LIBRARY
TECHNICAL REPORT SECTION
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

NPS55SS75072B

NAVAL POSTGRADUATE SCHOOL

Monterey, California

SYSTEM TEST METHODOLOGY VOL. II

by

G. H. BRADLEY
G. T. HOWARD
N. F. SCHNEIDEWIND
G. W. MONTGOMERY
T. F. GREEN

July 1975

Approved for public release; distribution unlimited

Prepared for:
val Air Development Center
rminster, Pennsylvania

FEDDOCS
D 208.14/2:NPS-55SS72072B

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Isham Linder
Superintendent

Jack R. Borsting
Provost

This work herein was supported by the Naval Air Development Center, Warminster, Pennsylvania.

Reproduction of all or part of this report is authorized.

Prepared by:

Table of Contents

Appendices

A. Description of Prototype Testing Simulation Model A-1

B. FORTRAN Example Program Code B-1

C. Directions for the Use of the Error Simulation Program C-1

D. Error Simulation Program Listing D-1

E. Directions for the use of the Analytical Error Detection Model E-1

APPENDIX A

DESCRIPTION OF PROTOTYPE TESTING SIMULATION MODEL

1. Definition of Terms

The following terms were defined in the text: state, module, task, and primary state. Their usage in this description of the simulation of the model will be consistent with these definitions. The status of a module was defined to be the information required to identify a particular module, and its state, that is executing a task for a given input. Thus, the status of a module identifies: module, state, task, and input number.

All time values in the program were based on the standard time unit. The standard time unit may be defined as the user desires. The only requirement is that the user must be consistent throughout the program and that all time values be integers.

2. Program Characteristics

The simulation of the model was an event store type of simulation. In this type of simulation the time is advanced each time an event occurs instead of on a fixed interval. This prevented the occurrence of a large number of intervals during which there was no activity. This type of simulation maintains the complete status of the system during the simulation because events are executed in the order in which they occur. The events in this simulation consisted of the arrival of inputs, the completion of a primary state, data collection events, and the termination event. The event that represented the completion of a primary state was always followed by the occurrence of the next primary state unless the input terminated or

the resources required to enter the next primary state were not available. A data collection event was a time period during which the user could measure any parameter of the simulation. These occurred at regular intervals throughout the running of the simulation and could be used to gather data to compute such statistics as average queue size.

The programming language chosen for the simulation language was FORTRAN IV. The reason for this choice was to ensure the portability of the program. FORTRAN is not an ideal programming language for simulation. Other languages such as GPSS (General Purpose Simulation System) provide many of the building blocks needed for the simulation such as the clock and the calendar. Although FORTRAN does not provide these blocks, it is readily available at most institutions. A limit to the portability of the program is the random number generator which was used in the simulation. The program used the LLRANDOM package available at the W. R. Church Computer Center. The random numbers were passed by subroutine calls. By adjusting these calls, a new random number generator may be introduced into the program.

The simulation was designed to be able to simulate a wide range of computer systems. This was implemented by allowing the user to input various characteristics of the system. These characteristics are discussed in the input section of the appendix. The resources used were identified throughout the program only by number. The user was to assign both the type of resources the system uses and the amount of each resource available to the system. The only requirement was that all quantities were to be integer values. The workload for the system was defined as a system of tasks. The user was required to specify the priority of tasks. Modules were identified by number. The user was to specify which module or modules was to execute each task. The user was also to specify which resources and primary states were required in the performance of each task.

The simulation was programmed in a modular fashion. Each major function of the simulation consisted of a separate subroutine. This design was chosen to facilitate modifications to the program and to improve the readability of the program. All major data structures and variables were made available to all subroutines by the use of common blocks.

3. Inputs

The simulation allowed the user to vary the input parameters listed below. The description is in the order in which the inputs were required by the program. If no range is given the parameter did not have limits placed upon it by the program. Limits were caused by the size of the data structure used in the program. The user could have chosen any reasonable value for the unlimited parameters. Subscripted variables indicate multi-dimensional data structures.

* MAXIN --- Determined the number of inputs the system may process at one time. Range was one to ten inputs.
* MEANIN --- Represented the mean interarrival time for the system. Units had to be standard time units.
* NUMRES --- Total number of resources required by the system for proper functioning of the system plus one. The additional resource was the system time representation. This allowed the user to specify the amount of time the resources were in use. The range of NUMRES was from two to ten.
* MAXRES(I) --- This vector gave the maximum amount of each resource available to the system. The units had to be in standard time units. The range of I was from one to NUMRES.

* NUMTSK --- This was the number of tasks assigned to the system. The range was from one to ten tasks.
* PRECTSK(I,J) --- This matrix was used to give the precedence relationships among tasks. Each column was identified by a task. This task blocks all tasks listed in the column. The first element of each column was the number of tasks blocked by the given task. For example, if column number three contained the following numbers reading from top down: 2, 4, 6. This would indicate task number three blocked two tasks, namely task four and task six. The variable I had a range from one to NUMTSK. J was confined to the same range.
* MAXSEQ --- This was the maximum number of primary states that a module had to go through in order to execute a task. The range was one to eight states.
* DOTSK(I,J) --- This matrix showed the different sequences of primary states that a module was required to go through to complete one task. Each row represented the path for one module to perform one task. The first element of a row was the task, the second element was the module, and the remaining elements were the sequence of primary states. A zero indicated the sequence had been completed. The variables I and J had a range of one to ten.

* ENDTR --- This was the number of different primary states for every module. The maximum value was thirty.
* TSKRES(I,J) --- This matrix gave the amount of each resource required for the execution of each primary state of a module and the associated task. Each row was the resources required by one task/state/module. Thus, the first three elements of each row represented the respective task, state, and module associated with the resource requirements given in that row. The first entry represented the time that the resources would be required in that state. The variable I had a range from one to ENDTR. The range of J was from one to NUMRES plus three.

The following parameters were assigned values in the start subroutine of the program and may be varied as the user desires:

* MAXTIM --- This variable gave the total time the simulation was to be run. Units were standard time units. The value assigned was 1000000.
* DATINV --- This variable was the time between data collection events. Units were standard time units. The value assigned was 2500. The user was free to choose any value for this parameter. However, the value chosen should be realistic in terms of the other time values used in the program.
* IX --- This variable was the seed for the random number generator. It could have had a value

The system table was represented by a group of vectors. A vector was a one dimensional array that was used to store status information. The group of vectors was accessed by a common index. The index was stored on a separate vector in order of request for resources. Thus, a complete description of any module waiting for resources could be obtained without a sequential search of the table. This facilitated the user specification of service discipline.

The simulation calendar was represented by a group of vectors in the same manner as the system queue. Besides the information describing the status of modules that were in service, this group also contained vectors representing simulation information such as event type and event time. The event time was the time representing the scheduled completion of the event. The event type differentiated among end of state events, data collection events, arrival of input events, and termination events. A zero in this vector represented an empty slot on the simulation calendar. Each time an event was selected the event time vector was searched sequentially to determine the next event occurrence time in the simulation.

The event records represented a diverse group of data structures used to preserve information about the events in the simulation. The general form of these data structures were matrices with each row representing one input that was being processed by the system and a column for each data element to be preserved for each state. The data elements were items such as time input entered system, time module used resources in a primary state, and the order in which primary states were executed. In order to avoid being overwritten, these matrices had to be printed out at the end of each sequence of tasks. The column index to these matrices was based on the number of state transitions and each index was adjusted for the proper number of

elements being preserved in the matrix.

Utility records were used to save information as it was being moved from the system queue to the simulation calendar.

5. Output

Two types of output were provided for in the simulation of the model. These were event reports and sequential reports. The event reports occurred whenever the event with which they were associated was terminated. Three types of event reports were used in the simulation. These were end of state report, end of task sequence report and termination report.

The end of state report report occurred at the end of every primary state. The report contained the status of the module. The report gave timing information about the primary state just completed. This included the time the module was placed in the table, the time the module entered service, time the module finished service, total time in the system, total time in service, and total time in table. The report showed the amount of each resource available after the resources used in that state were returned to the system. A printout of the system table was shown after the next module status had been added to the table. All modules that were scheduled at that time were reported giving the status of each module. The report concluded with the status of all system resources after the scheduling activity had been completed.

The end of task sequence report occurred after the complete sequence of tasks for one input had been completed. The first part of the report gave the sequence of states, identifying each state by task and module. The second part of the report gave information about the timing of the task

sequence. This included the time the first module entered service, the time the last module finished processing, the total time required if the sequence of tasks had been processed serially, and the total time the input was in the system.

The termination report gave the total number of inputs that were lost by the system when it was already processing the maximum number of inputs allowed in the system. This report could be used to give averages from the data collection events such as average queue size and average percent of each resource being used by the system.

Sequential reports were not used by the simulation but could have been initiated by the data collection event and used to give an instantaneous status of the system at regular intervals.

6. Program Flow

The program commenced with a call to subroutine START. This subroutine initialized all the variables and returned control to the main program. The next event was selected by the subroutine NEXT. This subroutine selected an event based on a sequential search of the simulation calendar. Control was returned to the main program which called the appropriate subroutine to handle the selected event.

If the next event was an arrival of an input, subroutine NEWINP was called. Subroutine NEWINP inserted the first module status on the system table with a call to the subroutine PUTONQ. This occurred only if there was space in the system for another input and if there was room in the system table. If there was no space in the system, the input was counted as a lost input. If there was no room on the system table, the program was terminated with an error condition. With the exception of an error condition, the next

arrival of an input was placed on the simulation calendar with a call to the subroutine ENTER. A call was placed to the subroutine SKED. This subroutine ascertained whether currently available system resources were sufficient for a module on the system table to commence service. All modules were examined on a first-come-first served basis and as many as possible were placed in service. The end of state event was entered on the simulation calendar. Control was passed to the main program where the next event was selected.

The subroutine ENDSTT was called. The resources used by the module were returned to the system. If the completion of the state did not complete the task, the next module status was placed on the table and subroutine SKED was called. If the completion of the state completed the task, the next task was selected. The first module status of the task was placed on the system table and again SKED was called. In either case, STTDUMP was called to produce the end of state report. If the task was a termination task, subroutine DUMPIN was called to produce an end of task sequence report. Control was returned to the main program.

Two other types of events could occur. These were the data collection event and the termination event. A data collection event resulted in a call to subroutine WHAT. This subroutine collected the assigned information, scheduled the next data collection event, and returned control to the main program. The termination event occurred at the maximum simulation time and produced the termination report.

7. Desirable Features

The generality of the simulation, and hence its applicability to any real system, was restricted because certain parameter choices are not given to the user. These parameters included the queueing discipline, resource allocation policy and the distribution of time spent in a state. Ideally, there would

be many choices of these parameters with the user having the ability to specify the discipline, policy, and distribution that best represented the system he intended to test.

During the development of the program, other features that would have added to the usefulness of the simulation were discovered. One such feature was a debug package. This could include such items as a stall alarm. This alarm would give a warning if a module was in the system table for an excessive time period. The length of this time period could be selected and adjusted by the user. This alarm could be used to detect modules that were deadlocked or to set a maximum time that a module could be in the table before a failure occurred. Beizer [7] indicated various other alarms that would appear useful. These include watchdog timer alarms, cycle alarms, improper sequence alarms, and multiple or missed interrupt alarms (if the user chooses to simulate interrupts).

Another feature that would be useful is a restart capability. The status of the system at the termination of the simulation could be saved in a dump file. The user would be allowed to stop the system and make any observations he desired and restart from the same point. This ability would be particularly useful in investigating errors that occurred in a random fashion.

The above feature could be best utilized as an interactive program. The tester could watch the display of pertinent system data as the simulation operated. When an unusual occurrence was observed, the tester could stop the simulation and query parameters to determine what had taken place. Having made his observations, the tester would have the choice of restarting the simulation, continuing from the point of interruption, or saving the data produced on hard copy.

8. Limitation

The simulation that was programmed had a limitation for practical

use in testing. The limitation was the size of the system that could be represented. The number of tasks and other elements were limited. This was done to reduce turn-around time during the development of the program.

APPENDIX B

FORTRAN EXAMPLE PROGRAM CODE

Node			Code
	C		SUBROUTINE FOR COMPUTING J-N, Y-N, I-N, AND K-N
	C		ASYMPTOTIC EXPANSION IS USED FOR LARGE ARGUMENT
	C		
1			SUBROUTINE BESSEL(X, NORD,BJN,BY,BIN,BK)
			PI = 3.14152927
			GAM = .57721566
	C		
			FN = NORD
2			IF (X - FN - 6.) 1,200,200
		1	XA = X / 2.
			XB = XA * XA
	C		
	C		COMPUTATION OF J-ZERO AND I-ZERO
	C		
			N = 0
	C		
	C		COMPUTATION OF J-N AND I-N BY POWER SERIES
	C		
		3	AN = N
			T = 1.
			S = -1.
4		10	IF(AN) 9999,20,12
		12	T = T * XA / AN
			AN = AN - 1.
7,11			GO TO 10
		20	BJN = T
			BIN = T
8			DO 30 K = 1,K
			DEN = K * (K + N)
			T = T * XB / DEN
12			IF ((BJN + T) - BJN) 25,50,25
		25	BJN = BJN + T*S
			BIN = BIN + T
15		30	S = -S
	C		
14	C		
	C		
17		50	IF (N - 1) 75,130,55
	C		
	C		K - N IS COMPUTED FROM ASYMPTOTIC EXPANSION
	C		IF X IS GREATER THAN N + 3
	C		
21		55	IF (X - FN - 3.) 1111,1111,200
	C		
	C		CALCULATION OF J-1 AND I-1
	C		
		65	N = 1
			BJO = BJN
			BIO = BIN
			BYO = BY
			BKO = BK
			GO TO 3

```
Node                Code
          C
          C         COMPUTATION OF K-ZERO AND Y-ZERO
          C
             75     BY  = 2./PI * (GAM + LOGF(XA)) * BJN
                    BK  = - (GAM + LOGF(XA)) * BIN
                    T = XB
                    S = 1.
                    XI = 1.
22                  DO  110 K = 2,K
                    AK = K
25                  IF  ((BY + T*XI) - BY) 100,120,100
                100 BK = BK + T*XI
                    BY = BY + 2./PI*S*T*XI
                    T = T*XB / (AK*AK)
                    XI = XI + 1./AK
28              110 S = -S
          C
27        C
          C
29              120  IF (NORD)  9999,55,65
          C
          C         COMPUTATION OF Y-1 AND K-1 BY WRONSKIAN FORMULAS
          C
                130 BY = (BJN*BYO - 2./(PI*X)) / BJO
                    BK =  (1. / x - BIN*BKO) / BIO
          C
          C         Y-N AND K-N BY RECURSION FORMULAS
          C         FOR ORDERS HIGHER THAN ONE
          C
                    P = 1.
20                  IF (NORD  - 1) 9999,55,140
                140 BY1 = BY
                    BK1 = BK
                    BY  = 2.*P / X*BY1 - BYO
                    BK  = 2.*P / X*BY1 + BYO
          C
                    P = P + 1.
          C
23                  IF (NORD - 2)  9999,150,146
                146 BYO = BY1
                    BK = BK1
                    NORD = NORD - 1
                    GO TO 140
          C
                150 N = P
                    GO TO 3
          C
```

Node		Code
	C	
	C	COMPUTATION OF J-N, I-N, K-N, AND Y-N
	C	BY ASYMPTOTIC EXPANSIONS
	C	
	200	C = 4 * NORD * NORD
		D = 8. * X
		CCN1 = SQRTF(2./(PI*X))
		CON2 = 1./SQRTF(2.*PI*X)
		CON3 = SQRTF(PI/(2.*X))
		AN = NORD
		PHI = X - (2.*AN + 1.) / 4. * PI
		M = X + 1. + SQRTF(X * X + AN * AN)
		T = (C - 1.) / D
		S = 1.
		U = 1.
		PN = 1.
		QN = T
		BK = 1. + T
		BI = 1. - T
	C	
3		DO 240 I = 2,M
		AI = I
		T = (C - (2.*AI - 1.)**2) / D*T / AI
		BK = BK + T
		BI = BI + T*S
5		IF (S) 220,9999,210
	210	PN = PN - T*S*U
		U = -U
		GO TO 230
10		IF ((QN + T) - QN) 230,241,230
	220	QN = QN - T*S*U
	230	S = -S
12	240	CONTINUE
13	241	BK = EXPF(-X)*CON3*BK
	C	
	C	ASYMPTOTIC EXPANSION IS USED ONLY FOR K-N
	C	IF X IS BETWEEN N + 3 AND N + 6
	C	
16		IF (X - FN -6.) 1111,250,250
	250	BJN = CON1 * (PN*COSF(PHI) - QN*SINF(PHI))
		BY = CON1 * (PN*COSF(PHI) + QN*SINF(PHI))
		BIN = EXPF(X) * CON2 * BI
	C	
18,24	1111	NORD = FN
		RETURN
	C	
6,9,19, 26,30	9999	STOP
		END

APPENDIX C

DIRECTIONS FOR USE OF THE ERROR SIMULATION PROGRAM

Random numbers are drawn using the Naval Postgraduate School Random Number Generator Package LLRANDOM (NPS55LW73061A). These calls are RANDOM for a uniform distribution and EXPON for an exponential distribution.

Printer plots are drawn by the PLOTP routine which is part of the IBM supplied routines. Directed graph plots are drawn by the DRAWP routine available on the Naval Postgraduate School IBM 360/67 System connected to a CALCOMP plotter. The subroutine GRAPHO is limited because it requires the directed graph to be constructed in such a way that the rightmost leg of the tree is the longest leg, which means that the rightmost terminal node is the lowest node.

Histograms are plotted and analyzed by the HISTG routine available on the Naval Postgraduate School IBM 360/67. The functions SIN, COS and ATAN2 are standard IBM FORTRAN-supplied subroutines.

The analysis of loops in the simulation assumed that there were instructions in the arc from i to j and also in the backward arc from j to i. This may not be realistic, but does prove useful when simulating the actual path. When an input reaches a branch point that has a backward arc emanating from it, the model simulates a random number of iterations using the number of instructions in the backward

arc. After completing the simulated looping, it was possible to trace the same path or a different path if there were intermediate branch points. If the same path was selected, there was another chance for the backward arc to be selected. This is a common programming structure, where a loop is executed for so many iterations, a few parameters are changed and the loop is executed again for another or the same number of iterations.

The comment section of the simulation program, titled "Directions for Use", describes the use of each data input to the program as well as the common values utilized during this research. The numbers along the top of the cards (Figures C-1 to C-7) are the column numbers, the middle row of numbers are sample data and the bottom row of numbers is either the corresponding node for each piece of data or data item names. The formats used are shown below.

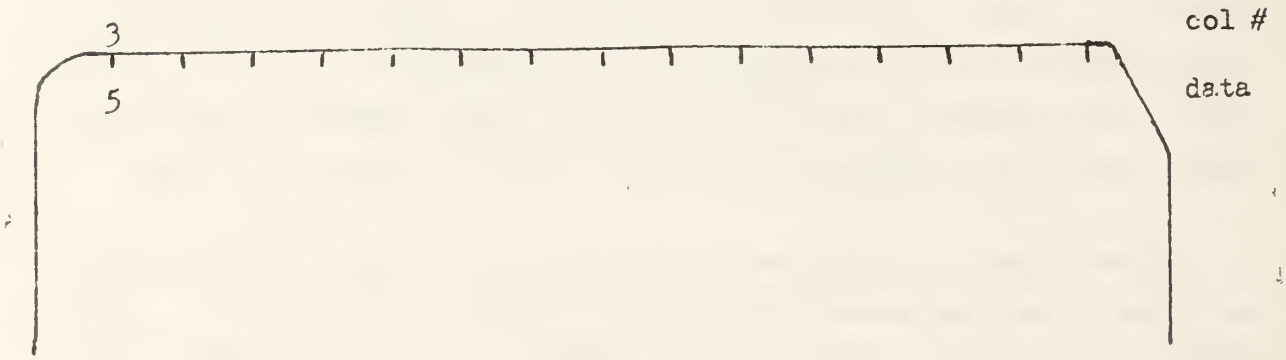


Fig C-1 Each item in paragraph A of the directions for use is on a separate card with I3 format except item A(12) which is A5 format (*****).

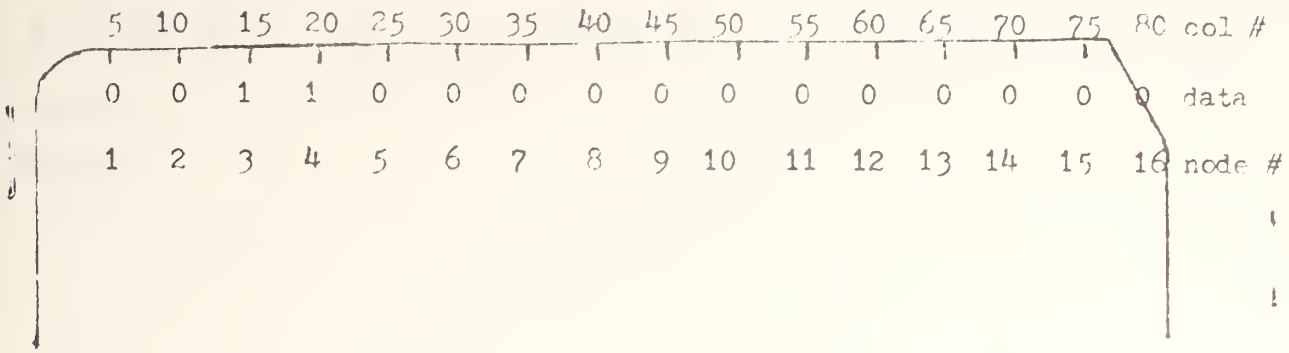


Fig C-2 Adjacency matrix card - 16I5 format.

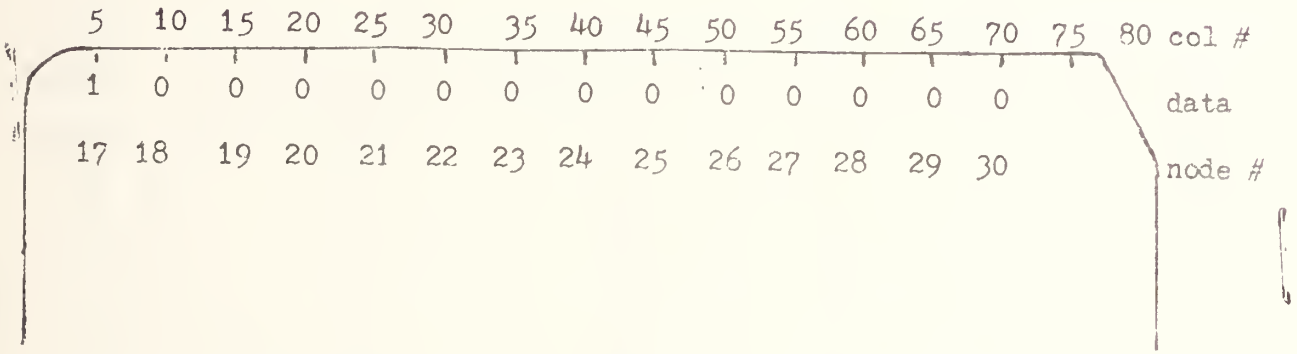


Fig C-3 Second card of adjacency matrix - based on a graph with 30 nodes.

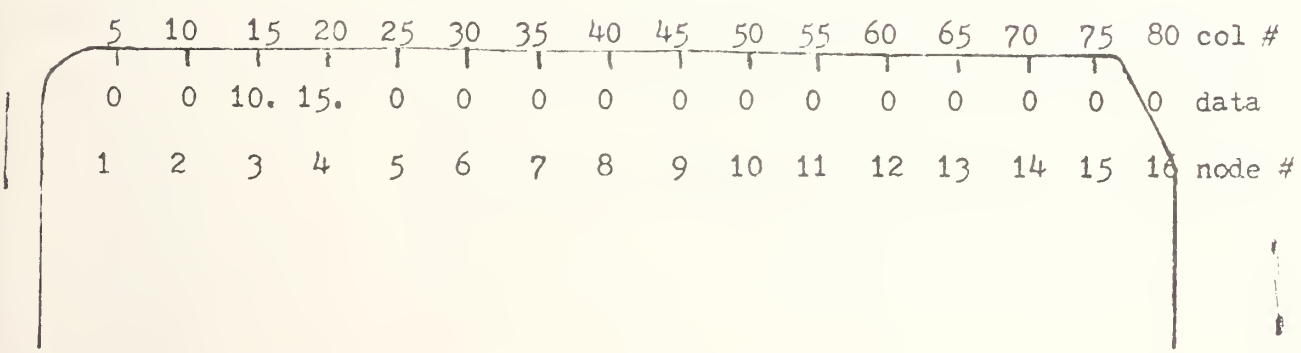


Fig C-4 Matrix of arc lengths data card - 16F5.0 format.

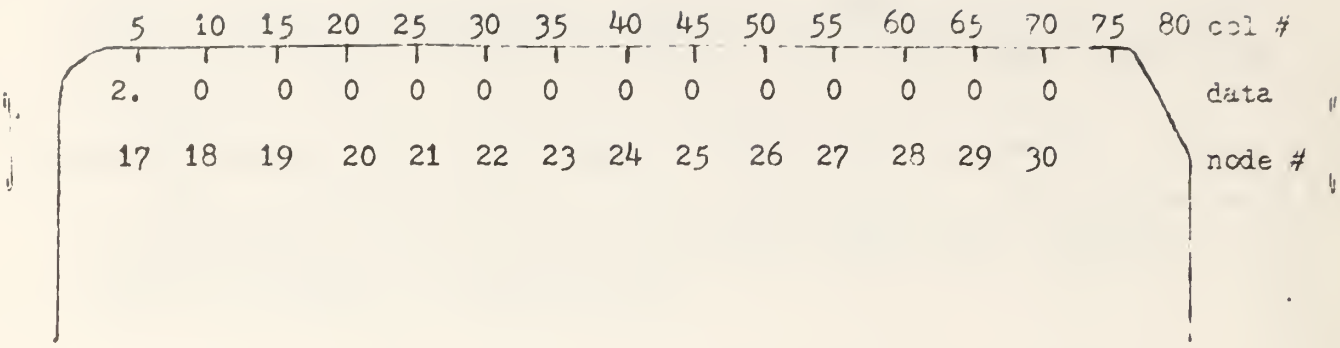


Fig C-5 Second card of the matrix of arc lengths - based on a graph with 30 nodes.

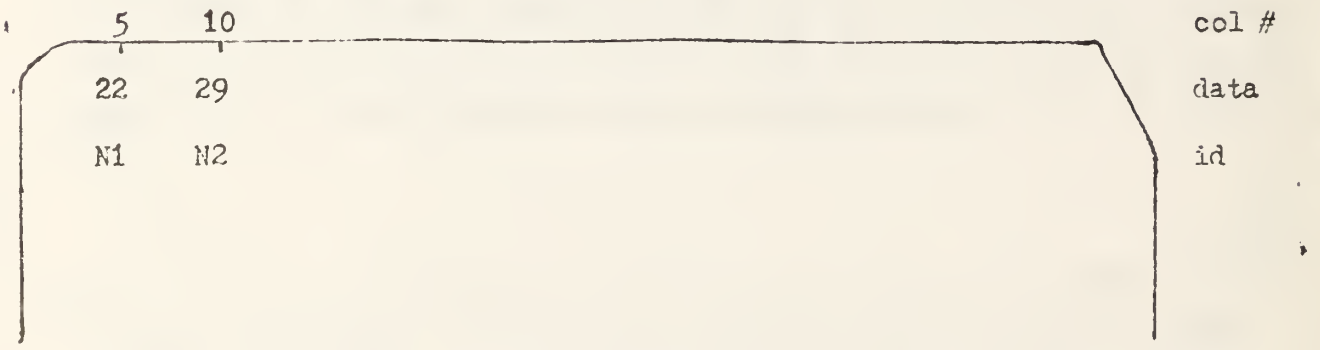


Fig C-6 The two nodes which determine the arc to be added or deleted from the structure - 2I5 format

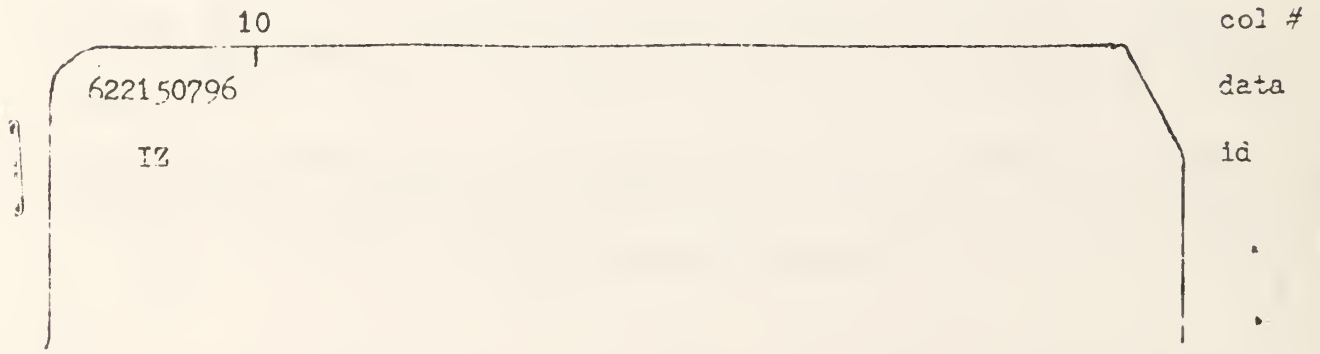


Fig C-7 The seed for the random number generator - format 1110.

```
//GREEN05 JOB (1723,0409,CS42),'GREEN SMC 2827',TIME=4,MSGCLASS=0
// EXEC FORTCLGP,REGION.GO=200K
//FORT.SYSPRINT DD SYSOUT=0
//FORT.SYSIN DD *
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                             C
C      SIMULATION OF ERRORS IN SOFTWARE                                       C
C                                                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                             C
C                                                                             C
C                              DIRECTIONS FOR USE                             C
C                                                                             C
C     1. TO RUN THE SIMULATION WHICH VARIES THE NUMBER OF INSTRUCTIONS        C
C        BETWEEN EACH NODE.                                                   C
C        A. NORMAL DATA INCLUDES:                                             C
C           (1). SIMNUM = 1                                                   C
C           (2). MINPUT = 3                                                   C
C           (3). MEANLN = 1                                                   C
C           (4). NUMOUT = 30                                                  C
C           (5). INREAD = 0                                                   C
C           (6). DELADD = 0                                                   C
C           (7). ITIME = 50    (IN MILLISECONDS)                              C
C           (8). MMTTR = 30    (IN MINUTES)                                   C
C           (9). MEANER = 20                                                  C
C          (10). MEANIT = 20                                                  C
C          (11). AVCHNG = 3                                                   C
C          (12). ASTER = *****                                                C
C          (13). N = 30                                                       C
C        B. INPUT ADJACENCY MATRIX                                            C
C                                                                             C
C     2. TO RUN THE SIMULATION WHICH VARIES THE NUMBER OF INPUTS USED:        C
C        A. NORMAL DATA INCLUDES:                                             C
C           (1). SIMNUM = 2                                                   C
C           (2). MINPUT = 1                                                   C
C           (3). MEANLN = 10                                                  C
C           (4). NUMOUT = 30                                                  C
C           (5). INREAD = 0  IF THE NUMBER OF INSTRUCTIONS IN EACH            C
C                            ARC IS TO BE DETERMINED RANDOMLY - OTHERWISE     C
C                            INREAD = 1                                       C
C           (6). DELADD = 0                                                   C
C           (7). ITIME = 50    (IN MILLISECONDS)                              C
C           (8). MMTTR = 30    (IN MINUTES)                                   C
C           (9). MEANER = 20                                                  C
C          (10). MEANIT = 20                                                  C
C          (11). AVCHNG = 3                                                   C
C          (12). ASTER = *****                                                C
C          (13). N = 30                                                       C
```

```
C       B. INPUT ADJACENCY MATRIX                                          C
C       C. IF INREAD = 1  INPUT MATRIX OF ARC LENGTHS.                     C
C                                                                          C
C    3. TO RUN THE SIMULATION WHICH VARIES THE NUMBER OF ARCS BETWEEN      C
C       THE NODES.                                                         C
C       A. NORMAL DATA INCLUDES:                                           C
C          (1). SIMNUM = 3                                                 C
C          (2). MINPUT = 3                                                 C
C          (3). MEANLN = 10                                                C
C          (4). NUMOUT = 20                                                C
C          (5). INREAD = 0                                                 C
C          (6). DELADD = 0   IF DELETING ARCS FROM STRUCTURE.              C
C               DELADD = 1   IF ADDING ARCS TO THE STRUCTURE.              C
C          (7). ITIME = 50    (IN MILLISECONDS)                            C
C          (8). MMTTR = 30    (IN MINUTES)                                 C
C          (9). MEANER = 20                                                C
C         (10). MEANIT = 20                                                C
C         (11). AVCHNG = 3                                                 C
C         (12). ASTER = *****                                              C
C         (13). N = 30                                                     C
C       B. IF DELADD = 0  INPUT MOST COMPLEX STRUCTURED ADJACENCY          C
C          MATRIX.                                                         C
C          IF DELADD = 1  INPUT SIMPLEST STRUCTURED ADJACENCY MATRIX.      C
C       C. INPUT 20 (NUMOUT) PAIRS OF VALUES N1 AND N2 WHICH               C
C          DETERMINE THE ARC TO BE DELETED OR ADDED.                       C
C                                                                          C
C    4. TO RUN THE SIMULATION WHICH VARIES THE NUMBER OF LOOPS:            C
C       A. NORMAL DATA INCLUDES:                                           C
C          (1). SIMNUM = 4                                                 C
C          (2). MINPUT = 3                                                 C
C          (3). MEANLN = 10                                                C
C          (4). NUMOUT = 20                                                C
C          (5). INREAD = 0                                                 C
C          (6). DELADD = 0   IF DELETING LOOPS FROM STRUCTURE.             C
C               DELADD = 1   IF ADDING LOOPS TO THE STRUCTURE.             C
C          (7). ITIME = 50    (IN MILLISECONDS)                            C
C          (8). MMTTR = 30    (IN MINUTES)                                 C
C          (9). MEANER = 20                                                C
C         (10). MEANIT = 20                                                C
C         (11). AVCHNG = 3                                                 C
C         (12). ASTER = *****                                              C
C         (13). N = 30                                                     C
C       B. IF DELADD = 0  INPUT MOST COMPLEX STRUCTURED ADJACENCY          C
C          MATRIX.                                                         C
C          IF DELADD = 1  INPUT SIMPLEST STRUCTURED ADJACENCY MATRIX.      C
C       C. INPUT 20 (NUMOUT) PAIRS OF VALUES N1 AND N2 WHICH               C
C          DETERMINE THE LOOP TO BE DELETED OR ADDED.                      C
C                                                                          C
```

```
C      5. TO RUN THE SIMULATION WHICH VARIES THE SEED TO THE INPUT:                C
C         A. NORMAL DATA INCLUDES:                                                 C
C              (1). SIMNUM = 5                                                     C
C              (2). MINPUT = 1                                                     C
C              (3). MEANLN = 10                                                    C
C              (4). NUMOUT = 30                                                    C
C              (5). INREAD = 0   IF THE NUMBER OF INSTRUCTIONS IN EACH             C
C                   ARC IS TO BE DETERMINED RANDOMLY - OTHERWISE                   C
C                   INREAD = 1                                                     C
C              (6). DELADD = 0                                                     C
C              (7). ITIME = 50    (IN MILLISECONDS)                                C
C              (8). MMTTR = 30    (IN MINUTES)                                     C
C              (9). MEANER = 20                                                    C
C             (10). MEANIT = 20                                                    C
C             (11). AVCHNG = 3                                                     C
C             (12). ASTER = *****                                                  C
C             (13). N = 30                                                         C
C         B. INPUT ADJACENCY MATRIX                                                C
C         C. IF INREAD = 1  INPUT MATRIX OF ARC LENGTHS.                           C
C         D. IZ = 1722632  (OR WHATEVER LAST SEED WAS ON PREVIOUS RUN)             C
C            NOTE:  LAST DATA CARD                                                 C
C                                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON/ALLL/N, NODES(30,30)
      COMMON/ARRAY/X(30,30), NUMPTS(30), ASTOUT(30), AVCHNG
      COMMON/ERROR/ISEED(30,30), ITER(30,30), MEANER, INST, MEANIT, IZ
      COMMON/OUT/NSEED, INPUT, SVSEED(30,30)
      COMMON/GEN/MEANLN, IX
      COMMON/NEW/IW
      COMMON/GRAB/INREAD
      DIMENSION ITIME(50), TREP(50), TLINK(50), TKOUNT(50), TINPUT(50)
      DIMENSION PSEED(50), F(50), FNA(50), FLA(50)
      DIMENSION TSEED(50), TINST(50), PTEST(50)
      INTEGER*4 CHANGE, AVCHNG, TESTED(30,30), TEST, SIMNUM, SVSEED
      INTEGER*4 DELADD
      REAL*4 MTTR
  100 FORMAT (I3)
  108 FORMAT (I10)
  140 FORMAT (1615)
  151 FORMAT ('1', 13X, 'THIS RUN WILL VARY THE NUMBER OF INSTRUCTIONS B
     1ETWEEN EACH NODE.')
  152 FORMAT ('1', 13X, 'THIS RUN WILL VARY THE NUMBER OF INPUTS TO BE U
     1SED.')
  153 FORMAT ('1', 13X, 'THIS RUN WILL VARY THE NUMBER OF ARCS BETWEEN T
     1HE NODES.')
  154 FORMAT ('1', 13X, 'THIS RUN WILL VARY THE NUMBER OF LOOPS.')
  155 FORMAT ('1', 13X, 'THIS RUN WILL VARY THE RANDOM NUMBER SEED WHICH
     1 DETERMINES THE INPUT PATH.')
```

```
156 FORMAT ('0', 13X, 'THE NUMBER OF INSTRUCTIONS IN EACH ARC HAS BEEN
   1 READ IN AND IS NOT RANDOM.')
180 FORMAT ('0', 13X, 'ADJACENCY MATRIX'//7X, 30(I4))
203 FORMAT ('0', 13X, 'NOTE: NONEXISTENCE OF AN ARC IS INDICATED BY A
   1 ZERO'//)
204 FORMAT ('0', 13X, 'THERE WILL BE', I3, ' GRAPHS EVALUATED THIS RUN
   1.'//)
205 FORMAT (7(I5, 5X))
206 FORMAT (A5)
207 FORMAT ('0', 12X, 'INPUT ASSUMPTIONS (VARIABLES):'//13X, 'EXECUTIO
   1N TIME PER INSTRUCTION HAS AN EXPONENTIAL DISTRIBUTION WITH A MEAN
   2 OF', I3,' MILLISECONDS.'//13X, 'MEAN TIME TO REPAIR AN ERROR HAS
   3AN EXPONENTIAL DISTRIBUTION WITH A MEAN OF', I3,' MINUTES.'//13X,
   4'THE NUMBER OF INPUT PATHS BEING INVESTIGATED IS', I3, '.'//13X,
   5'ERRORS ARE SEEDED BY AN EXPONENTIAL DISTRIBUTION WITH A MEAN OF',
   6 I3,' INSTRUCTIONS PER ERROR.'//13X, 'THE AVERAGE NUMBER OF INSTRU
   7CTIONS CHANGED WHEN AN ERROR IS FOUND IS', I3,'.')
208 FORMAT ('0', 13X, 'THE NUMBER OF ITERATIONS PER LOOP IS UNIFORMLY
   1DISTRIBUTED FROM 1 TO', I3, '.')
209 FORMAT ('0', 13X, 'THE MEAN NUMBER OF INSTRUCTIONS BETWEEN EACH NO
   1DE IS EXPONENTIALLY DISTRIBUTED WITH A MEAN OF', I3, '.')
227 FORMAT ('0', 13X, 'SAMPLE INPUT PATH FROM'//13X, 'NODE TO NODE'//)
230 FORMAT('0', 11X, I4, 4X, I4)
240 FORMAT('0',13X,'FOR INPUT NUMBER',I3,' THE EXECUTION TIME IS',
   1 F8.2,     ' SECONDS WITH A TOTAL REPAIR TIME OF', F8.2, ' HOURS'
   2 // 13X, 'AND AN AVERAGE LINK TRAVERSAL TIME OF', F8.2,' SECONDS')
245 FORMAT ('0', 13X, 'ERROR FOUND'//)
250 FORMAT('1', 13X, I4, ' SEEDED ERRORS REMAINING'//13X, 'MATRIX OF
   1SEEDED ERRORS REMAINING'//7X, 30(I4))
255 FORMAT(' ', 7X, 30(A4))
260 FORMAT('0',     I5, ' *', 30(I4))
265 FORMAT('1', 11X, I3, ' INPUTS'//13X, I4, ' INSTRUCTIONS'//13X,
   1I3, ' ERRORS SEEDED'//13X, I3, ' RESIDUAL ERRORS'//13X, F6.2,
   2 ' PERCENT RESIDUAL ERRORS'//13X, F6.2, ' PERCENT OF ARCS TESTED'
   3 //)
270  FORMAT('0',  9X, F10.5, ' SECONDS EXECUTION TIME'//12X,
   1 F10.5, ' HOURS TO REPAIR ERRORS'      //12X, F10.6, ' SECONDS AVE
   2RAGE ARC TRAVERSAL TIME'//)
275  FORMAT('0', 12X, F5.2, ' IS THE RATIO OF ACTUAL TO MAXIMUM NUMBER
   1 OF ARCS'// 13X, F5.2, ' IS THE RATIO OF NODES TO ARCS'//13X,
   2 F5.2, ' IS THE RATIO OF LOOPS TO ARCS'//)
280 FORMAT ('0'//13X, 'HISTOGRAM FOR EXECUTION TIME')
281 FORMAT ('0'//13X, 'HISTOGRAM FOR REPAIR TIME')
282 FORMAT ('0'//13X, 'HISTOGRAM FOR PERCENT RESIDUAL ERRORS')
283 FORMAT ('0'//13X, 'HISTOGRAM FOR PERCENT ARCS TESTED')
284 FORMAT ('1', 1X)
285 FORMAT ('0'//13X,'EXECUTION TIME VS RESIDUAL ERRORS')
286 FORMAT ('0'//13X,'EXECUTION TIME VS PERCENT RESIDUAL ERRORS')
```

```
  287 FORMAT ('0'//13X, 'REPAIR TIME VS RESIDUAL ERRORS')
  288 FORMAT ('0'//13X, 'NUMBER OF INPUTS VS RESIDUAL ERRORS')
  289 FORMAT ('0'//13X, 'NUMBER OF INSTRUCTIONS VS RESIDUAL ERRORS')
  290 FORMAT ('0'//13X, 'RATIO OF ACTUAL TO MAXIMUM NUMBER OF ARCS VS RE
     1SIDUAL ERRORS')
  291 FORMAT ('0'//13X, 'RATIO OF NODES TO ARCS VS RESIDUAL ERRORS')
  292 FORMAT ('0'//13X, 'RATIO OF LOOPS TO ARCS VS RESIDUAL ERRORS')
  293 FORMAT ('0'//13X,'EXECUTION TIME VS RATIO OF ACTUAL TO MAXIMUM NUM
     1BER OF ARCS')
  294 FORMAT ('0'//13X,'EXECUTION TIME VS RATIO OF NODES TO ARCS')
  295 FORMAT ('0'//13X,'EXECUTION TIME VS RATIO OF LOOPS TO ARCS')
  296 FORMAT ('0'//13X, 'NUMBER OF INPUTS VS PERCENT OF ARCS TESTED')
  297 FORMAT ('0'//13X, 'RATIO OF ACTUAL TO MAXIMUM NUMBER OF ARCS VS PE
     1RCENT OF ARCS TESTED')
  298 FORMAT ('0'//13X, 'RATIO OF NODES TO ARCS VS PERCENT OF ARCS TESTE
     1D')
  299 FORMAT ('0'//13X, 'RATIO OF LOOPS TO ARCS VS PERCENT OF ARCS TESTE
     1D')
  300 FORMAT ('0'//13X, 'NUMBER OF INPUTS VS EXECUTION TIME')
  301 FORMAT ('0'//13X, 'NUMBER OF INPUTS VS PERCENT RESIDUAL ERRORS')
  302 FORMAT ('0'//13X,'EXECUTION TIME VS NUMBER OF INSTRUCTIONS')
  303 FORMAT ('1', 59X, 'DATA SUMMARY '//62X, 'PERCENT', 3X, 'PERCENT',
     1  2X, 'EXECUTION', 3X, 'REPAIR', 4X, 'ACTUAL', 4X, 'NODES', 5X,
     2  'LOOPS'/4X, 'RUN', 35X, 'ERRORS', 3X, 'RESIDUAL', 2X, 'RESIDUAL',
     3  4X, 'ARCS', 6X, 'TIME', 6X, 'TIME', 5X, 'TO MAX', 6X, 'TO', 8X,
     4  'TO'/2X,
     5  'NUMBER', 5X, 'NODES', 4X, 'INPUTS', 4X, 'INSTR.', 4X, 'SEEDED',
     6  4X, 'ERRORS', 4X, 'ERRORS', 4X, 'TESTED', 4X,  '(SEC)', 7X,
     7  '(HRS)', 4X, 'ARCS', 6X, 'ARCS', 6X, 'ARCS'/)
  304 FORMAT ('0', 2X, I3, 1X, '*', 5X, I3, 7X, F5.0, 3(5X, F5.0),
     1  2(4X, F6.2), 2X, 2F10.5, 3X, F5.3, 2(5X, F5.3))
  305 FORMAT ('0', 13X, 'THE LAST SEED USED WAS', I11, '.')
C       READ IN THE TYPE OF SIMULATION TO BE RUN (SIMNUM)
        READ (5,100) SIMNUM
C       READ IN NUMBER OF INPUTS TO BE USED (MINPUT)
        READ (5,100) MINPUT
C       READ IN THE MEAN NUMBER OF INSTRUCTIONS BETWEEN EACH NODE (MEANLN)
        READ (5,100) MEANLN
C       READ IN THE NUMBER OF GRAPHS TO BE EVALUATED (NUMOUT)
        READ (5,100) NUMOUT
C       READ IN WHETHER OR NOT THE NUMBER OF INSTRUCTIONS IS TO BE READ IN
        READ (5,100)  INREAD
C       READ IN WHETHER DELETING AN ARC FROM THE STRUCTURE OR
C       ADDING AN ARC TO THE STRUCTURE.
        READ (5,100)  DELADD
C       READ IN THE MEAN TIME TO EXECUTE AN INSTRUCTION IN MILLISECONDS
C       (ITIME)
        READ (5,100) ITIME
```

```
C       READ IN MEAN TIME TO REPAIR AN ERROR IN MIN (MMTTR)
        READ (5,100) MMTTR
C       READ IN MEAN NUMBER OF INSTRUCTIONS THAT ARE ERROR FREE (MEANER)
        READ (5,100) MEANER
C       READ IN MEAN NUMBER OF ITERATIONS FOR EACH LOOP (MEANIT)
        READ (5,100) MEANIT
C       READ IN MEAN NUMBER OF INSTRUCTIONS CHANGED WHEN AN ERROR IS
C       FOUND (AVCHNG)
        READ (5,100) AVCHNG
C       READ IN THE ASTERICKS TO PUT AROUND OUTPUT ARRAYS (ASTER)
        READ (5,206) ASTER
C       READ IN THE NUMBER OF NODES (N)
        READ (5,100) N
C       READ IN THE ADJACENCY MATRIX (NODES)
         DO 20 I=1,N
   20    READ (5,140)  (NODES(I,J), J=1,N)
        IF (SIMNUM.EQ.1)    WRITE (6,151)
        IF (SIMNUM.EQ.2)    WRITE (6,152)
        IF (SIMNUM.EQ.3)    WRITE (6,153)
        IF (SIMNUM.EQ.4)    WRITE (6,154)
        IF (SIMNUM.EQ.5)    WRITE (6,155)
        IF (INREAD.EQ.1)  WRITE(6,156)
        WRITE (6,207) ITIME, MMTTR, MINPUT, MEANER, AVCHNG
        WRITE (6,208) MEANIT
        WRITE (6,209) MEANLN
        WRITE (6,204) NUMOUT
        NMOUT = 1
        IX = 71286223
        IW = IX
C       DETERMINE THE MAXIMUM NUMBER OF ARCS IN THE ADJACENCY MATRIX
        MAXARC = N*(N-1)
C       NUMPTS IS THE LABEL FOR THE OUTPUT ARRAY
        DO 57 I = 1,N
        NUMPTS(I) = I
C       ASTER AND ASTOUT ARE TO PUT ASTERICKS AROUND THE ARRAY
        ASTOUT(I) = ASTER
   57 CONTINUE
        CALL DIGRAF
   59 DO 62 I = 1,N
        DO 61 J = 1,N
C       ISEED IS THE ARRAY OF ERRORS SEEDED
        ISEED(I,J) = 0
   61 CONTINUE
   62 CONTINUE
C        SEED THE PROGRAM WITH ERRORS
        IF (SIMNUM.EQ.1)   IX = 1722632
        CALL SEED
C       THIS SEED CAN BE CHANGED TO THE LAST VALUE OF THE SEED ON THE
```

```
C       PREVIOUS TEST IF A CONTINUATION OF THE DATA IS DESIRED.
        IF (SIMNUM.EQ.5)  READ (5,108)  IX
   63   ARCS = 0.
        INST = 0
        NSEED = 0
        DO 65 I=1,N
        DO 65 J=1,N
C       ZERO THE MATRIX WHICH RECORDS WHETHER OR NOT AN ARC HAS BEEN TEST
        TESTED(I,J) = 0
        IF (NODES(I,J).NE.1)  GO TO 65
C       CALCULATE THE NUMBER OF ARCS IN THE PROGRAM
        ARCS = ARCS + 1.
C       CALCULATE THE NUMBER OF INSTRUCTIONS
        INST = INST + X(I,J)
C       CALCULATE THE NUMBER OF ERRORS SEEDED
        NSEED = NSEED + ISEED(I,J)
   65   CONTINUE
C       CALCULATE THE NUMBER OF LOOPS IN THE PROGRAM
        LOOPS = 0
        DO 68 I=1,N
        DO 68 J=1,I
        IF (NODES(I,J).EQ.1)  LOOPS = LOOPS + 1
   68   CONTINUE
C       CHOOSE SAMPLE INPUT
        INPUT = 1
   77   TOTREP = 0.
        TIME = 0.
        AVLINK = 0.
        TTIME(NNOUT) = 0.
        TREP(NNOUT) = 0.
        TLINK(NNOUT) = 0.
   78   IZ = IX
        NODE = 1
        WRITE (6,227)
C       NUMSUC IS THE NUMBER OF SUCCESSORS
   79   NUMSUC = 0
        DO 80 J=1,N
        IF (NODES(NODE,J).EQ.1) NUMSUC = NUMSUC + 1
   80   CONTINUE
        IF (NUMSUC.EQ.0) GO TO 96
        IF (NUMSUC.NE.1) GO TO 82
        K = 1
        GO TO 83
C       THE SUCCESSORS ARE CHOSEN RANDOMLY WITH A UNIFORM DISTRIBUTION
   82   CALL RANDOM(IZ, U, 1)
        K = 1 + NUMSUC*U
   83   KK = 0
        DO 85 J = 1,N
```

```
      IF (NODES(NODE,J).EQ.1) KK = KK + 1
      IF (KK.EQ.K) GO TO 86
   85 CONTINUE
   86 L = J
C     TIME IS ASSUMED TO HAVE A MEAN OF ITIME MILLISECONDS PER
C       INSTRUCTION WITH  AN EXPONENTIAL DISTRIBUTION
      CALL EXPON(IZ, XTIME, 1)
      XTIME = XTIME * ITIME
      IF (NODE.LE.L) GO TO 88
C     CONVERT TIME INTO HOURS
      TIME = TIME + XTIME * ITER(NODE,L) * (X(NODE,L) + X(L,NODE)) /
     1 1.0E03
      GO TO 89
   88 TIME = TIME + X(NODE,L) * XTIME/  1.0E03
   89 WRITE (6,230) NODE,L
      TESTED(NODE,L) = 1
C     IS THERE AN ERROR IN THIS PATH
      IF(ISEED(NODE,L).EQ.0) GO TO 95
C     ERROR FOUND
      WRITE (6,245)
C     SEE IF A NEW ERROR SHOULD BE INSERTED
      CALL NUSEED
   90 ISEED(NODE,L) = ISEED(NODE,L) - 1
      CALL  EXPON(IZ, U, 1)
C     MEAN TIME TO REPAIR HAS AN EXPONENTIAL DISTRIBUTION WITH A
C       MEAN OF MMTTR MINUTES
      MTTR = MMTTR * U
C     TOTREP IS THE TOTAL REPAIR TIME IN HOURS
      TOTREP = TOTREP + MTTR/ 60.
      GO TO 78
   95 NODE = L
      GO TO 79
C     AVLINK IS THE AVERAGE ARC  TRAVERSAL TIME IN MINUTES
   96 AVLINK = (TIME/ARCS)
      WRITE (6,240) INPUT, TIME, TOTREP, AVLINK
      IF (SIMNUM.EQ.2)  GO TO 97
      TTIME(NNOUT) = TTIME(NNOUT) + TIME
      TREP(NNOUT) = TREP(NNOUT) + TOTREP
      TLINK(NNOUT) = TLINK(NNOUT) + AVLINK
      GO TO 99
   97 NOUT = NNOUT - 1
      IF (NOUT.GT.1)  GO TO 98
      TLINK(1) = AVLINK
      TREP(1) = TOTREP
      TTIME(1) = TIME
      GO TO 99
   98 TLINK(NNOUT) = TLINK(NOUT)  + AVLINK
      TREP(NNOUT) = TREP(NOUT)  + TOTREP
```

```
      TTIME(NNOUT) = TTIME(NOUT)  + TIME
   99 TIME = 0.
      TOTREP = 0.
      AVLINK = 0.
      INPUT = INPUT + 1
C     KOUNT IS A COUNTER OF THE NUMBER OF SEEDED ERRORS REMAINING
      KOUNT = 0
      DO 101 I = 1,N
      DO 101 J = 1,N
      IF (NODES(I,J).EQ.1) KOUNT = KOUNT + ISEED(I,J)
  101 CONTINUE
      WRITE (6,250) KOUNT, (NUMPTS(I), I = 1,N)
      WRITE (6,255) (ASTOUT(I), I = 1,N)
      DO 110 I = 1,N
  110 WRITE (6,260) NUMPTS(I), (ISEED(I,J), J=1,N)
C     DISTINCT SEED FOR RANDOM NUMBER GENERATOR FOR EACH INPUT
      IX = IX - 12345
      IF (SIMNUM.EQ.2)  GO TO 111
      IF (SIMNUM.EQ.5)  GO TO 111
C     MINPUT IS THE MAXIMUM NUMBER OF INPUT PATHS TO BE CHECKED
      IF (INPUT.LE.MINPUT) GO TO 78
C     CALCULATE THE NUMBER OF ARCS TESTED
  111 TEST = 0
      DO 112 I = 1,N
      DO 112 J = 1,N
      IF (TESTED(I,J).EQ.1)  TEST = TEST + 1
  112 CONTINUE
      INPUT = INPUT - 1
C     PTEST    IS THE PERCENT OF ARCS TESTED
      PTEST(NNOUT) = 100. * TEST / ARCS
C     F IS THE RATIO OF ACTUAL TO MAXIMUM NUMBER OF ARCS
      F(NNOUT) = ARCS / MAXARC
C     FNA IS THE RATIO OF NODES TO ARCS
      FNA(NNOUT) = N / ARCS
C     FLA IS THE RATIO OF LOOPS TO ARCS
      FLA(NNOUT) = LOOPS / ARCS
      TINST(NNOUT) = INST
      TINPUT(NNOUT) = INPUT
      TSEED(NNOUT) = NSEED
      TKOUNT(NNOUT) = KOUNT
      PSEED(NNOUT) = 100. * (TKOUNT(NNOUT) / NSEED)
      WRITE (6,265) INPUT, INST, NSEED, KOUNT, PSEED(NNOUT),
     1 PTEST(NNOUT)
      WRITE (6,270) TTIME(NNOUT), TREP(NNOUT), TLINK(NNOUT)
      WRITE (6,275) F(NNOUT), FNA(NNOUT), FLA(NNOUT)
      NNOUT = NNOUT + 1
      IF (NNOUT.GT.NUMOUT) GO TO 118
      IF (SIMNUM.EQ.1)  MEANLN = MEANLN + 1
```

```
      IF (SIMNUM.EQ.1)  GO TO 57
      IF (SIMNUM.EQ.2)  INPUT = INPUT + 1
      IF (SIMNUM.EQ.2)  GO TO 78
      DO 113  I = 1,N
      DO 113  J = 1,N
  113 ISEED(I,J) = SVSEED(I,J)
      IF (SIMNUM.EQ.5)  GO TO 63
  115 IF (DELADD.EQ.0)  CALL DELARC
      IF (DELADD.EQ.1)  CALL ADDARC
  116  WRITE (6,180) (NUMPTS(I), I=1,N)
      WRITE (6,255) (ASTOUT(I), I = 1,N)
      DO 117 I = 1,N
  117  WRITE (6,260)  NUMPTS(I), (NODES(I,J), J=1,N)
       WRITE (6,203)
      GO TO 63
  118 WRITE (6,305) IZ
      WRITE (6,284)
      CALL PLOTP (TTIME,  TKOUNT, NUMOUT, 0)
      WRITE (6,285)
      WRITE (6,284)
      CALL PLOTP (TTIME,  PSEED,  NUMOUT, 0)
      WRITE (6,286)
      WRITE (6,284)
      CALL PLOTP (TREP,   TKOUNT, NUMOUT, 0)
      WRITE (6,287)
      IF (SIMNUM.NE.2)  GO TO 119
      WRITE (6,284)
      CALL PLOTP (TINPUT, TKOUNT, NUMOUT, 0)
      WRITE (6,288)
      WRITE (6,284)
      CALL PLOTP (TINPUT,  PTEST,   NUMOUT, 0)
      WRITE (6,296)
      WRITE (6,284)
      CALL PLOTP (TINPUT, TTIME,    NUMOUT, 0)
      WRITE (6,300)
      WRITE (6,284)
      CALL PLOTP (TINPUT, PSEED,    NUMOUT, 0)
      WRITE (6,301)
  119 IF ((SIMNUM.EQ.2).OR.(SIMNUM.EQ.5))  GO TO 121
      WRITE (6,284)
      CALL PLOTP (TTIME,  TINST,    NUMOUT, 0)
      WRITE (6,302)
      WRITE (6,284)
      CALL PLOTP (TINST,  TKOUNT, NUMOUT, 0)
      WRITE (6,289)
      IF (SIMNUM.EQ.1)  GO TO 121
      WRITE (6,284)
      CALL PLOTP (F,      TKOUNT, NUMOUT, 0)
```

```
      WRITE (6,290)
      WRITE (6,284)
      CALL PLOTP (FNA,     TKOUNT, NUMOUT, 0)
      WRITE (6,291)
      WRITE (6,284)
      CALL PLOTP (FLA,     TKOUNT, NUMOUT, 0)
      WRITE (6,292)
      WRITE (6,284)
      CALL PLOTP (TTIME,   F,      NUMOUT, 0)
      WRITE (6,293)
      WRITE (6,284)
      CALL PLOTP (TTIME,   FNA,    NUMOUT, 0)
      WRITE (6,294)
      WRITE (6,284)
      CALL PLOTP (TTIME,   FLA,    NUMOUT, 0)
      WRITE (6,295)
      WRITE (6,284)
      CALL PLOTP (F,        PTEST,   NUMOUT, 0)
      WRITE (6,297)
      WRITE (6,284)
      CALL PLOTP (FNA,      PTEST,   NUMOUT, 0)
      WRITE (6,298)
      WRITE (6,284)
      CALL PLOTP (FLA,      PTEST,   NUMOUT, 0)
      WRITE (6,299)
  121 DO 125  K = 1,2
      WRITE (6,303)
      WRITE (6,255)  (ASTOUT(I), I = 1,N)
      DO 122  I = 1,NUMOUT
  122 WRITE (6,304) I, N, TINPUT(I), TINST(I), TSEED(I), TKOUNT(I),
     1 PSEED(I), PTEST(I), TTIME(I), TREP(I), F(I), FNA(I), FLA(I)
  125 CONTINUE
      IF ((SIMNUM.NE.3).AND.(SIMNUM.NE.4))  CALL GRAPHO
C     NEED A MINIMUM OF 10 GRAPHS TO ANALYZE BEFORE CALLING HISTG
      IF (SIMNUM.NE.5)  GO TO 126
      CALL HISTG (TTIME, NUMOUT, 0)
      WRITE (6,280)
      CALL HISTG (TREP,  NUMOUT, 0)
      WRITE (6,281)
      CALL HISTG (PSEED, NUMOUT, 0)
      WRITE (6,282)
      CALL HISTG (PTEST, NUMOUT, 0)
      WRITE (6,283)
  126 CONTINUE
C
      STOP
C
      END
```

```
      SUBROUTINE DIGRAF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C     CONSTRUCT THE DIRECTED GRAPH                                             C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON/ALLL/N, NODES(30,30)
      COMMON/ARRAY/X(30,30), NUMPTS(30), ASTOUT(30), AVCHNG
      COMMON/GEN/MEANLN, IX
      COMMON/ERROR/ISEED(30,30), ITER(30,30), MEANER, INST, MEANIT, IZ
      COMMON/GRAF/INREAD
      INTEGER*4 CHANGE, AVCHNG
  100 FORMAT (16F5.0)
  130 FORMAT ('0', 10X, I4, 1X, 'NODES IN DIRECTED GRAPH.'//)
  140 FORMAT('0', 13X, 'MATRIX OF ARC LENGTHS'//7X, 30(I4))
  150 FORMAT ('0',      I5, ' *', 30(F4.0))
  160 FORMAT ('0', 13X, 'NOTE: 0.0 INDICATES NO ARC  FROM I TO J'//)
  180 FORMAT ('0', 13X, 'ADJACENCY MATRIX'//7X, 30(I4))
  190 FORMAT ('0',      I5, ' *', 30(I4))
  200 FORMAT ('0', 13X, 'NOTE: NONEXISTENCE OF AN ARC IS INDICATED BY A
     1 ZERO'//)
  210 FORMAT (' ', 7X, 30(A4))
  222 FORMAT (' ', 7X, 30(A4))
  225 FORMAT ('0',      I5, ' *', 30(I4))
  230 FORMAT ('0', 13X, 'LOOP ITERATIONS'//7X, 30(I4))
C     IF INREAD = 0 RANDOMLY GENERATE INSTRUCTION LENGTHS
C     IF INREAD = 1 READ IN INSTRUCTION LENGTHS
      IF (INREAD.EQ.0)  GO TO 20
      DO 10  I = I,N
   10 READ (5,100)  (X(I,J), J = 1,N)
      GO TO 25
C     GENERATE THE NUMBER OF INSTRUCTIONS BETWEEN EACH NODE -
C        EXPONENTIALLY DISTRIBUTED
   20 DO 25 I=1,N
      DO 25 J=1,N
      X(I,J) = 0.0
      IF (NODES(I,J).EQ.0) GO TO 25
      CALL EXPON (IX,XLNTH, 1)
C     MEANLN IS THE EXPONENTIALLY DISTRIBUTED MEAN LENGTH OF EACH ARC
      X(I,J) = MEANLN * XLNTH + 1.0
   25 CONTINUE
      WRITE (6,130) N
       WRITE (6,140) (NUMPTS(I), I = 1,N)
      WRITE (6,210) (ASTOUT(I), I = 1,N)
       DO 30 I=1,N
   30  WRITE (6,150)  NUMPTS(I), (X(I,J), J=1,N)
       WRITE (6,160)
       WRITE (6,130) (NUMPTS(I), I=1,N)
```

```
      WRITE (6,210) (ASTOUT(I), I = 1,N)
       DO 50 I=1,N
   50  WRITE (6,190)  NUMPTS(I), (NODES(I,J), J=1,N)
       WRITE (6,200)
C     THE NUMBER OF ITERATIONS PER LOOP IS DISTRIBUTED WITH A UNIFORM
C     DISTRIBUTION WITH A MEAN OF MEANIT
      DO 55 I = 1,N
      DO 55 J = 1,N
      ITER(I,J) = 0
      IF (NODES(I,J).EQ.0) GO TO 55
      ITER(I,J) = 1
      IF (J.GT.I) GO TO 55
      CALL RANDOM (IX, TITER, 1)
      ITER(I,J) = 1 + MEANIT*TITER
   55 CONTINUE
C     CREATE LABELS FOR THE OUTPUT ARRAY
      WRITE (6,230)  (NUMPTS(I), I=1,N)
      WRITE (6,222) (ASTOUT(I), I=1,N)
      DO 73 I = 1,N
   73 WRITE (6,225) NUMPTS(I), (ITER(I,J), J=1,N)
      RETURN
      END
C
C
C
C
C
      SUBROUTINE ADDARC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C     ADDARC ADDS AN ARC BETWEEN TWO NODES                            C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON/ALLL/N, NODES(30,30)
      COMMON/ARRAY/X(30,30), NUMPTS(30), ASTOUT(30), AVCHNG
      COMMON/GEN/MEANLN, IX
  100 FORMAT (2I5)
  110 FORMAT ('1', 13X, 'A NEW ARC HAS BEEN ADDED FROM NODE', I3,
     1   ' TO NODE', I3, ' OF LENGTH', F4.0)
C     ADD A NEW ARC FROM N1 TO N2
      READ (5,100) N1, N2
      NODES(N1,N2) = 1
      CALL EXPON (IX, XLNTH, 1)
C     MEANLN IS THE MEAN LENGTH OF THE ARC TO BE EXPONENTIALLY DISTRIB.
      X(N1,N2) = MEANLN * XLNTH + 1.0
      WRITE (6,110) N1, N2, X(N1,N2)
      RETURN
      END
```

```
      SUBROUTINE DELARC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     DELARC DELETES AN ARC BETWEEN TWO NODES                          C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON/ALLL/N, NODES(30,30)
      COMMON/ARRAY/X(30,30), NUMPTS(30), ASTOUT(30), AVCHNG
      COMMON/ERROR/ISEED(30,30), ITER(30,30), MEANER, INST, MEANIT, IZ
      COMMON/OUT/NSEED, INPUT, SVSEED(30,30)
      INTEGER SVSEED
  100 FORMAT (2I5)
  110 FORMAT ('1', 13X, 'AN ARC WAS DELETED FROM NODE', I3,
     1  ' TO NODE', I3, '.'/)
      READ (5,100) N1, N2
      NODES(N1,N2) = 0
      X(N1, N2) = 0.0
      ISEED(N1,N2) = 0
      SVSEED(N1, N2) = 0
      WRITE (6,110) N1, N2
      RETURN
      END
C
CCC
C
      SUBROUTINE SEED
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     SEED THE GRAPH WITH ERRORS                                       C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON/ALLL/N, NODES(30,30)
      COMMON/ARRAY/X(30,30), NUMPTS(30), ASTOUT(30), AVCHNG
      COMMON/ERROR/ISEED(30,30), ITER(30,30), MEANER, INST, MEANIT, IZ
      COMMON/OUT/NSEED, INPUT, SVSEED(30,30)
      COMMON/GEN/MEANLN, IX
      INTEGER*4 CHANGE, AVCHNG, SVSEED
      DIMENSION ER(30), IER(30)
  205 FORMAT ('0', 13X, 'SEED ERRORS AT INSTRUCTION INTERVALS'// 10X,
     1 30I4)
  210 FORMAT ('1'// 13X, 'TOTAL NUMBER OF INSTRUCTIONS IS', I5)
  220 FORMAT ('0'// 13X, 'TOTAL NUMBER OF ERRORS SEEDED IS',I4// 13X,
     1'THE ERROR MATRIX'//7X, 30(I4))
  222 FORMAT (' ', 7X, 30(A4))
  225 FORMAT ('0',     I5, ' *', 30(I4))
C
C     GENERATE ERRORS WITH EXPONENTIAL DISTRIBUTION
      CALL EXPON(IX, ER, N)
```

```
C       SPREAD ERRORS TO GIVE MEAN OF MEANER
C
        DO 60 I = 1,N
        IER(I) = MEANER * ER(I)
   60 CONTINUE
        II = 1
C       NSEED  IS THE NUMBER OF ERRORS SEEDED
        NSEED = 0
C       INST IS THE NUMBER OF INSTRUCTIONS
        INST = 0
C       NUMER IS THE ERROR SEED/INSTRUCTION COMPARATOR
        NUMER = 0
        DO 70 I = 1,N
        DO 65 J = 1,N
        IF (NODES(I,J).EQ.0)  GO TO 65
C       COUNT THE NUMBER OF INSTRUCTIONS
        INST = INST + X(I,J)
C       LOOPS ARE ASSUMED TO HAVE AN ERROR RATE PROPERTIONAL TO THE
C       NUMBER OF INSTRUCTIONS IN THE LOOP AND THE NUMBER OF ITERATIONS
C       THROUGH THE LOOP
        IF (J.LE.I) IER(II) = IER(II) / ( (X(I,J) * ITER(I,J))/MEANER)
   63 NUMER = NUMER + IER(II)
        IF (INST.LT.NUMER) GO TO 64
        ISEED(I,J) = ISEED(I,J) + 1
        NSEED = NSEED + 1
        II = II + I
        IF (II.LE.N)  GO TO 63
        II = 1
        GO TO 63
   64 NUMER = NUMER - IER(II)
   65 CONTINUE
   70 CONTINUE
        DO 71 I = 1,N
        DO 71 J = 1,N
C       SVSEED IS THE SAVE COPY OF THE ORIGINAL SEEDED ERRORS
   71 SVSEED(I,J) = ISEED(I,J)
C       NSEED  IS THE NUMBER OF ERRORS SEEDED
        WRITE (6,210) INST
        WRITE (6,205) (IER(I), I = 1,N)
        WRITE (6,220) NSEED, (NUMPTS(I), I = 1,N)
        WRITE (6,222) (ASTOUT(I), I=1,N)
        DO 75 I = 1,N
        WRITE (6,225) NUMPTS(I), (ISEED(I,J), J=1,N)
   75 CONTINUE
C
        RETURN
C
        END
```

```
      SUBROUTINE NUSEED
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C     CORRECTING PREVIOUS ERROR MAY CREATE A NEW ERROR                C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON/ALLL/N, NODES(30,30)
      COMMON/ARRAY/X(30,30), NUMPTS(30), ASTOUT(30), AVCHNG
      COMMON/ERROR/ISEED(30,30), ITER(30,30), MEANER, INST, MEANIT, IZ
      COMMON/OUT/NSEED, INPUT, SVSEED(30,30)
      COMMON/NEW/IW
      INTEGER*4 CHANGE, AVCHNG
      DIMENSION ERR(2)
  247 FORMAT ('0', 13X, 'NEW ERROR INSERTED FROM NODE', I4,' TO NODE',
     1 I4//)
      NUM = 0
C     THE ERROR IS EQUALLY LIKELY TO RESULT IN CHANGES TO ALL ARCS
      CALL RANDOM (IW, ERR, 2)
      I = 1 + N*ERR(1)
      J = 1 + N*ERR(2)
C     NEW ERROR CREATED AT (I,J)
C     IS THIS A NODE OF THE DIRECTED GRAPH
   10 IF (NODES(I,J).EQ.1) GO TO 20
      J = J + 1
      IF (J.GT.N) J = 1
      NUM = NUM + 1
C     IF NODE I IS A TERMINAL NODE NO NEW ERROR IS INSERTED
      IF (NUM.GT.N)  RETURN
      GO TO 10
C     CHANCE OF INSERTING A NEW ERROR IS BASED ON THE NUMBER OF
C     INSTRUCTIONS THAT MUST BE CHANGED
   20 CALL RANDOM (IW, ERNU, 1)
C     AVCHNG IS THE ASSUMED (INPUT) AVERAGE NUMBER OF INSTRUCTIONS
C     CHANGED WHEN AN ERROR IS FOUND
C     CHANGE IS THE UNIFORMLY DISTRIBUTED NUMBER OF INSTRUCTIONS CHANGED
      CHANGE = 1 + AVCHNG*ERNU*2.
C     CRATIO IS THE RATIO OF INSTRUCTIONS CHANGED TO THE NUMBER OF
C     INSTRUCTIONS IN THE UNIFORMLY RANDOMLY CHOSEN ARC
      CRATIO = CHANGE / X(I,J)
C     THE CRITERIA FOR INSERTING A NEW ERROR IS UNIFORMLY DISTRIBUTED
      CALL RANDOM (IW, RATIO, 1)
C     IF THE CHANGE RATIO (CRATIO) IS GREATER THAN THE CRITERIA (RATIO),
C     THEN A NEW ERROR IS INSERTED
      IF (CRATIO.LE.RATIO) RETURN
   50 ISEED(I,J) = ISEED(I,J) + 1
      NSEED = NSEED + 1
      WRITE (6,247)  I,J
      RETURN
```

```
      END
      SUBROUTINE GRAPHO
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     GRAPHICALLY DISPLAY THE DIRECTED GRAPH USING THE CALCOMP PLOTTER C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON/ALLL/N, NODES(30,30)
      DIMENSION ND(30,30), X(30), XX(2), Y(30), YY(2)
      DIMENSION XL(36), YL(36), THETA(36)
      INTEGER*4 ITB(12)/12*0/
      REAL*4 RTB(28)/28*0.0/
C     ITB IS DRAWP ARRAY
      ITB(1) = 1
      ITB(2) = 0
      ITB(3) = 9
      ITB(4) = 15
      ITB(8) = 2
      ITB(9) = 0
      ITB(10) = 2
      ITB(11) = 0
C     RTB IS DRAWP ARRAY
      RTB(1) = 1.0
      RTB(2) = 1.0
      DO 4 I = 1,N
      DO 4 J = 1,N
C     ND IS ARRAY USED TO CHECK FOR PREDECESSORS
      ND(I,J) = 0
    4 CONTINUE
      X(1) = 4.5
      Y(1) = 15.0
      X(2) = X(1)
C     CONST  IS Y-DIRECTION SCALE FACTOR
      CONST = 30./N
      Y(2) = Y(1) - CONST
      CALL DRAWP(2,X,Y,ITB,RTB)
      XCONST = 1.5
C     KOUNT  IS X-DIRECTION SHRINK FACTOR - SHRINKS AFTER EVERY OTHER
C     FANOUT
      KOUNT = 1
C     LEVEL IS THE LEVEL IN THE DIRECTED GRAPH
      LEVEL = 2
      NN = N - 1
      ITB(1) = 2
C     LABEL IS THE NUMBER OF EACH NODE
      LABEL = 2
      DO 5 M = 1,N
      IF (NODES(1,M).EQ.0) GO TO 5
```

```
C       KK IS THE LAST NODE IN THE GIVEN LEVEL
        KK = M
        KT = KK + 1
        IF (NODES(1,KT).EQ.0) GO TO 6
     5  CONTINUE
C       KM IS THE LAST NODE IN THE NEXT LEVEL
     6  DO 7 M = KK,N
        IF (NODES(KK,M).EQ.0) GO TO 7
        KM = M
        KX = KM + 1
        IF (NODES(KK,KX).EQ.0) GO TO 8
     7  CONTINUE
     8  DO 50 I = 2,NN
        II = I - 1
C       NUMSUC IS THE NUMBER OF SUCCESSORS
        NUMSUC = 0
        DO 20 J = I,KM
        IF (NODES(I,J).EQ.0) GO TO 20
        DO 18 K = 1,II
C       IS THERE A NODE ALREADY IN EXISTENCE
        IF (ND(K,J).EQ.0) GO TO 18
C       CONNECT NEW NODE TERMINATING AT EXISTING NODE
    17  XX(1) = X(I)
        XX(2) = X(J)
        YY(1) = Y(I)
        YY(2) = Y(J)
        CALL DRAWP(2,XX,YY,ITB,RTB)
        ND(I,J) = 1
        GO TO 20
    18  CONTINUE
        NUMSUC = NUMSUC + 1
    20  CONTINUE
        IF (NUMSUC.LE.0) GO TO 44
    25  IF (NUMSUC.NE.1) GO TO 30
C       FANOUT TO ONE NEW NODE
        L = LABEL + 1
         LABEL = LABEL + 1
        ND(I,L) = 1
        X(L) = X(I)
        Y(L) = Y(I) - CONST
        XX(1) = X(I)
        XX(2) = X(L)
        YY(1) = Y(I)
        YY(2) = Y(L)
        CALL DRAWP(2,XX,YY,ITB,RTB)
C       IS THERE A NODE ABOVE THAT NEEDS CONNECTING TO THIS NODE
        DO 28 K = 1,II
        IF (ND(K,L).EQ.1) GO TO 28
```

```
      IF (NODES(K,L).EQ.0) GO TO 28
      XX(1) = X(K)
      YY(1) = Y(K)
      XX(2) = X(L)
      YY(2) = Y(L)
      ND(K,L) = 1
      CALL DRAWP(2,XX,YY,ITB,RTB)
   28 CONTINUE
      GO TO 44
C     FANOUT TO MULTIPLE NODES
   30 FAN = 2.*XCONST / (NUMSUC - 1)
      DO 35 L=1,NUMSUC
      M = LABEL + L
      ND(I,M) = 1
      IF (L.GT.1) GO TO 31
      X(M) = X(I) - XCONST
      GO TO 33
   31 LM = M - 1
      X(M) = X(LM) + FAN
   33 Y(M) = Y(I) - CONST
      XX(1) = X(I)
      YY(1) = Y(I)
      XX(2) = X(M)
      YY(2) = Y(M)
      CALL DRAWP(2,XX,YY,ITB,RTB)
C     IS THERE A NODE ABOVE THAT NEEDS CONNECTING TO THIS NODE
      DO 34 K=1,II
      IF (ND(K,M).EQ.1) GO TO 34
      IF(NODES(K,M).EQ.0) GO TO 34
      XX(1) = X(K)
      YY(1) = Y(K)
      XX(2) = X(M)
      YY(2) = Y(M)
      ND(K,M) = 1
      CALL DRAWP(2,XX,YY,ITB,RTB)
   34 CONTINUE
   35 CONTINUE
      LABEL = LABEL + NUMSUC
      KOUNT = KOUNT + 1
   40 IF (KOUNT.LT.2) GO TO 44
C     REDUCE THE X-DIRECTION SCALE FACTOR
      XCONST = XCONST/1.5
      KOUNT = 0
C     IS THIS NODE STILL ON THE GIVEN LEVEL
   44 IF (I.LT.KK) GO TO 50
C     NO, CHANGE LEVELS
      LEVEL = LEVEL + 1
C     DETERMINE THE LAST NODE IN THE NEW LEVEL
```

```
      DO 45 M = I,N
      IF (NODES(I,M).EQ.0) GO TO 45
      KK = M
      KT = KK + 1
      IF (NODES(I,KT).EQ.0) GO TO 46
   45 CONTINUE
C     DETERMINE THE LAST NODE IN THE NEXT LEVEL AFTER THE PRESENT LEVEL
   46 DO 47 M = KK,N
      IF (NODES(KK,M).EQ.0) GO TO 47
      KM = M
      KX = KM + 1
      IF (NODES(KK,KX).EQ.0) GO TO 50
   47 CONTINUE
   50 CONTINUE
C     GRAPH LOOPS
      DO 90 I = 2,N
      K = I - 1
      DO 80 J = 1,K
      IF (NODES(I,J).EQ.0) GO TO 80
      PI = 3.1415926
C     ESTABLISH THE DIRECTION OF CURVE FOR LOOP
      IF (X(I).LT.X(J)) PI = -PI
C     STARTING POINT FOR CURVE
      XL(1) = X(I)
      YL(1) = Y(I)
C     ENDING POINT FOR THE CURVE
      XL(36) = X(J)
      YL(36) = Y(J)
C     XDIF AND YDIF ARE THE DIFFERENCE BETWEEN THE X AND Y VALUES RESP.
      XDIF = X(I) - X(J)
      YDIF = Y(I) - Y(J)
C     R IS THE RADIUS OF THE LOOP
      R = 0.5 * SQRT(XDIF**2 + YDIF**2)
C     (XC,YC) IS THE CENTER OF THE LOOP
      XC = (X(I) + X(J))/2.
      YC = (Y(I) + Y(J))/2.
C     THETE(1) IS THE INITIAL ANGLE
      THETA(1) = ATAN2(YDIF,XDIF)
C     CREATE 36 POINTS IN A SEMICIRCLE FORMING THE LOOP
      DO 70 L = 2,35
      M = L - 1
      THETA(L) = THETA(M) + PI/35.
      XL(L) = R * COS (THETA(L)) + XC
      YL(L) = R * SIN (THETA(L)) + YC
   70 CONTINUE
      CALL DRAWP(36,XL,YL,ITB,RTB)
   80 CONTINUE
   90 CONTINUE
```

```
C       DRAW DIAMONDS AT EACH NODE
        ITB(1) = 3
        ITB(2) = 4
        CALL DRAWP(N,X,Y,ITB,RTB)
        RETURN
        END
//LINK.SYSPRINT DD SYSOUT=O
//GO.FT06F001 DD SYSOUT=O,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=3325),
//   UNIT=(SYSOUT,SEP=FT05F001),SPACE=(CYL,(4,1))
//GO.SYSIN DD *
   5
   1
  10
  50
   0
   1
  50
  30
  20
  20
   3
*****
  30
```

```
    0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     1     1     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     1     1     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     1     1     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     1     1     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     1     1     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     1     1     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     1     0     0     0     0     0     0     0     0     0     1     1
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    1     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     1     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

APPENDIX E

OPERATING DESCRIPTION OF ANALYTICAL ERROR DETECTION PROGRAMS

The analytic error detection model has been implemented as two FORTRAN programs, TRAV and EXP. Given a directed graph representation of a computer program with a unique start node and a unique exit node, branch probabilities and the expected number of errors on each arc, the programs compute the expected number of errors detected by each of a sequence of inputs that begin at the start node and end at the exit node. For example, for the program represented by Figure 1 with the indicated branch probabilities and the expected number of errors for each arc of 0.6, the output of program EXP is the table of Figure 2 and the graph of Figure 3.

The analysis assumes that an input is made at the start node, the arcs of the graph are traversed with the choice of arc at each branch node made according to the branch probabilities (which should sum to one for each branch node) and the input stops when it reaches the exit node. If an arc is traversed by an input, it is assumed that all the errors on that arc are detected and immediately corrected with no new errors introduced.

The error detection table and graph (for example, Figures 2 and 3) indicate how easy or difficult it is to detect errors in the computer program that is represented by the graph. It is possible to see how the error detection process is affected by different error distributions and it is possible to get some idea of when to stop testing for a given distribution of errors.

The computation is divided into two programs; the first computes the traversal probabilities for the graph--this is done only once for each graph. The second program uses the output of the first program plus an estimate of the expected number of errors on each arc to compute the expected number of errors detected. Typical use involves using the second program repeatedly with

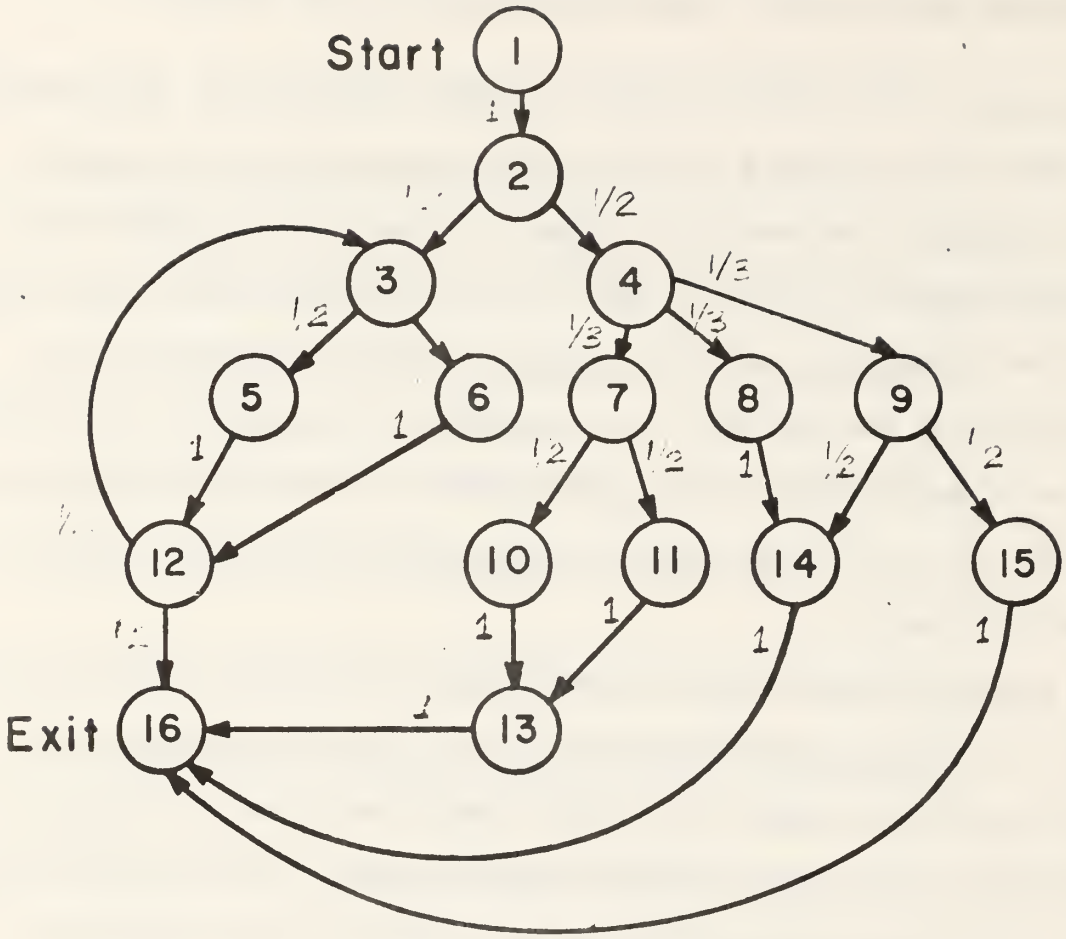

Start
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Exit
1
1/2
1/3
1/2
1/3
1/3
1
1
1/2
1/2
1
1/2
1/2
1
1
1
1
1
1

Figure 1

EXPECTED NUMBER OF ERRORS DETECTED VERSION 1.0

PROGRAM TITLE: ERROR DETECTION MODEL FIGURE 1

INPUTS = 20

NUMBER OF ARCS = 22

ARC TAIL	ARC HEAD	BRANCH PROB.	TRAVERSAL PROB.	EXPECTED NUMBER OF ERRORS
6	12	1.0000	0.3333	0.60
12	3	0.5000	0.2500	0.60
12	16	0.5000	0.5000	0.60
11	13	1.0000	0.0833	0.60
13	16	1.0000	0.1667	0.60
14	16	1.0000	0.2500	0.60
7	11	0.5000	0.0833	0.60
8	14	1.0000	0.1667	0.60
1	2	1.0000	1.0000	0.60
2	3	0.5000	0.5000	0.60
2	4	0.5000	0.5000	0.60
4	7	0.3333	0.1667	0.60
4	8	0.3333	0.1667	0.60
4	9	0.3333	0.1667	0.60
7	10	0.5000	0.0833	0.60
9	14	0.5000	0.0833	0.60
9	15	0.5000	0.0833	0.60
10	13	1.0000	0.0833	0.60
15	16	1.0000	0.0833	0.60
3	5	0.5000	0.3333	0.60
3	6	0.5000	0.3333	0.60
5	12	1.0000	0.3333	0.60

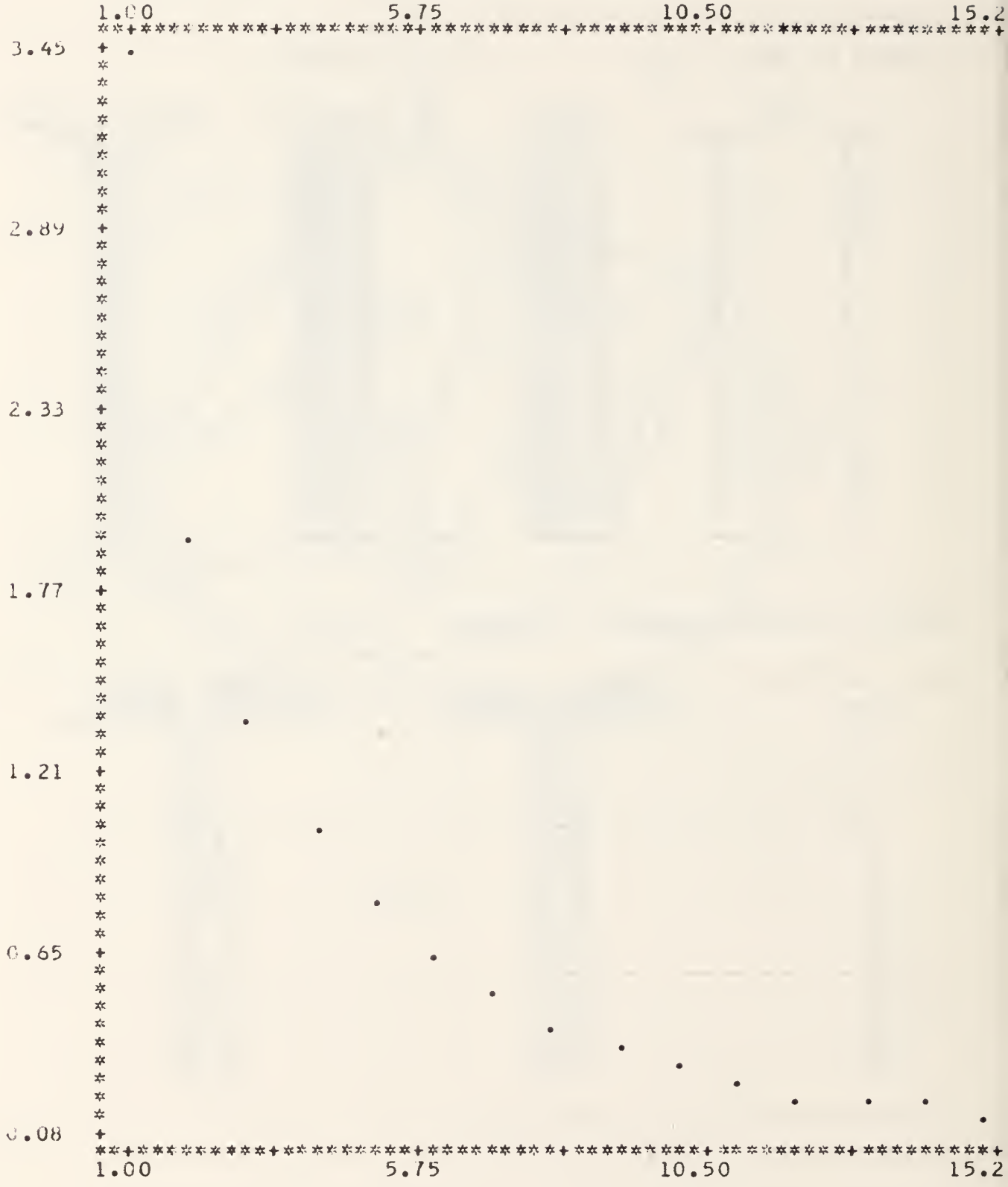
INITIAL EXPECTED NUMBER OF ERRORS= 13.20

INPUT	EXPECTED NUMBER OF ERRORS DETECTED	CUMMULATIVE EXPECTED NUMBER OF ERRORS DETECTED
1	3.45	3.45
2	1.95	5.40
3	1.39	6.79
4	1.04	7.82
5	0.80	8.62
6	0.63	9.25
7	0.51	9.76
8	0.42	10.19
9	0.36	10.54
10	0.30	10.85
11	0.26	11.10
12	0.22	11.33
13	0.20	11.52
14	0.17	11.69
15	0.15	11.84
16	0.13	11.98
17	0.12	12.10
18	0.11	12.20
19	0.09	12.30
20	0.08	12.38

Figure 2

GRAPH OF EXPECTED NUMBER OF ERRORS DETECTED -VS- NUMBER OF INPUTS.

PROGRAM TITLE: ERROR DETECTION MODEL FIGURE 1



X-SCALE: "*"= 0.237E 00 UNITS

Y-SCALE: "*"= 0.561E-01 UNITS

INPUT NUMBER

Figure 3

different error estimates. The compute time and storage requirements for the first program is many times that for the second program.

Program Trav

The first program, TRAV, computes for each arc the probability that an input beginning at the start node will traverse the arc. Since the first time an arc is traversed all errors are detected and corrected, it is not of interest if the arc is traversed one time or many times by a single input; thus the traversal probability is the probability of traversing the arc one or more times.

It is necessary that the user identify the arcs that can be traversed more than once by a given input; these arcs, called "repeated" arcs, must be treated differently in the computation. In the example, because arc $12 \rightarrow 3$ loops back, arcs $3 \rightarrow 5$, $5 \rightarrow 12$, $3 \rightarrow 6$, $6 \rightarrow 12$ and $12 \rightarrow 3$ are repeated arcs. (They are the only repeated arcs in Figure 1.) If a repeated arc is not identified by the user, the traversal probability will be wrong (too high). The traversal probability of a non-repeated arc that is erroneously identified as a repeated arc will be correct. However, since the calculation for repeated arcs involves inverting a matrix it saves computer time to correctly identify the repeated arcs.

For non-repeating arcs, the computation of the traversal probability is easy: The probability of reaching any node is the sum of the traversal probabilities of arcs coming into that node. The traversal probability of arcs leaving a node is the branch probability of that arc times the probability of reaching the node. For repeating arcs the calculation is more complex; for example, in Figure 1 the probability of reaching node 3 is 1/2 but the probability of traversing arc $3 \rightarrow 5$ is greater than 1/4 because even if an input traverses arc $3 \rightarrow 6$ there is some chance that node 3 will be revisited via arc $12 \rightarrow 3$ and arc $3 \rightarrow 5$ then traversed. The probability of traversing arcs $3 \rightarrow 6$, $6 \rightarrow 12$ and $12 \rightarrow 3$ and then traversing $3 \rightarrow 5$ for the first

time is $1/4 \cdot (1/2) \cdot (1/2) = 1/16$. The probability of traversing arc $3 \rightarrow 6$ twice then traversing arc $3 \rightarrow 5$ is $1/16(1/2)1/2 = 1/64$. Thus the traversal probability of arc $3 \rightarrow 5$ is $1/4 + 1/16 + 1/64 + \ldots = 1/3$. Program TRAV computes the traversal probability by means of a Markov chain analysis.

The input to program TRAV is the directed graph and its branch probabilities Each arc is input giving its tail node, its head node, its branch probability (i.e. probability of traversing the arc given you are at its tail node) and an indication if it is a repeated arc. Also input is the start and exit node numbers. The node numbers can be any integers between 01 and 99. There may be multiple arcs between the same two nodes. Since the sum of the branch probabilities from any node should equal one, the program checks for this. There should be no arcs branching from the exit node; the program checks for this also. The arcs can be input in any order.

Input format for program TRAV:

Card 1. (20A4) Alphanumeric title card.

Card 2. (1X,I2,1X,I2) Start node number and exit node number

Card 3 etc. (1X,I2,1X,I2,1X,A1,2X,F 10.5) for each arc of the graph, tail node number, head node number, alpha character 'R' indicates that the arc may be repeated, branch probability which is the probability that the arc will be traversed if you are at its tail node.

Last card. 00 or blanks in columns 2 and 3.

The output of the program is a printed and punched description of each arc with its branch and traversal probability. The punch output is ready (with some additions described below) for input to the second program.

Punch output of program TRAV:

Card 1. Title card

Card 2. Card to insert no. of inputs for expected errors program.

Card 3 etc. Same as input except for traversal probabilities added (F10.5) in columns 21 to 30.

Last card. 00 in columns 2 and 3.

Program TRAV is presently limited to 99 arcs. To accommodate more it is necessary to change "IDIM" and the arrays indicated in the comments cards of program TRAV and its subroutine MARKOV.

Data checks are made on the input and output:

Input check no. 1. Input format forces node numbers to be less than 100.

Input check no. 2. Check for input limit of 'IDIM' arcs.

Input check no. 3. Exit node should have no arcs branching away from it.

Input check no. 4. No branch probability should be less than zero or greater than one.

Input check no. 5. Check if there exist paths from start node that do not reach exit node.

Input check no. 6. Check that sum of branch probabilities from all nodes equal approximately one. If it does not, print warning and then divide by sum to make it equal to one.

Input check no 7. Check that start node is the tail of some arc and that exit node is the head of some arc.

The storage required depends on the number of arcs squared and the calculation of repeated arcs involves the inversion of a matrix that has a row for each arc. Thus storage and computer time can be reduced by decomposing a graph. For example, Figure 1 can be decomposed into two graphs, one with arcs $3 \rightarrow 5$, $3 \rightarrow 6$, $5 \rightarrow 12$, $6 \rightarrow 12$, $12 \rightarrow 16$ and $12 \rightarrow 3$ replaced by an arc $3 \rightarrow 16$ with branch probability 1 and the other with arcs $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 5$, $3 \rightarrow 6$, $6 \rightarrow 12$, $5 \rightarrow 12$, $12 \rightarrow 3$, $12 \rightarrow 16$ and an arc $2 \rightarrow 16$ with branch probability 1/2. The results of two applications of program TRAV can be combined to give the results for the original graph.

Computer time can also be saved by analyzing the graph before using the program. For example, in Figure 1, it is clear that the traversal probability of arcs $3 \rightarrow 6$, $3 \rightarrow 5$, $6 \rightarrow 12$ and $5 \rightarrow 12$ are equal. Thus only one need be marked "R" in the input (saving 3 inversions); the punch output can then be changed to correct the values for the 3 arcs without an "R."

Program EXP

The second program, EXP, computes the expected number of errors detected for a sequence of inputs. The input is the punched card output of the TRAV program, the expected number of errors for each arc and the number of inputs. The output is figures 2 and 3.

The calculation of the expected number of errors detected (and corrected) is straightforward. Let p_j be the traversal probability for arc j computed by the TRAV program and μ_j be the expected number of errors. The expected number of errors detected by the first input is $\sum p_j\mu_j$ where the summation is over all the arcs of the graph. After the first input the expected number of errors in arc j is $\mu_{j2} = \mu_j(1-p_j)$ (where the 2 indicates this is the expected number of errors before the second input). The expected number of errors detected by the second input is $\sum \mu_{j2}p_j$. In general, for the $k^{\underline{th}}$ input the expected number of errors is $\sum \mu_{jk}p_j$ where $\mu_{jk} = \mu_j(1-p)^{k-1}$.

Input format for program EXP:

Card 1. Title card output from traversal program.

Card 2. (I3) Number of inputs to the directed graph.

Card 3 etc. Traversal probability output from traversal program.

Card N. Final traversal card, 00 or blank in columns 2 and 3 output from traversal program.

Card N+1 etc. (8F10.2) Expected number of errors for each arc. Ordering same as traversal probability input above.

Last card. No final card is necessary.

The output is a table and a graph of the expected number of errors detected and a graph of the cumulative number of errors detected.

Program EXP is currently limited to 99 arcs just as program TRAV. It is also limited to 150 inputs, to accommodate more inputs it is necessary to change "LIMIT" and the arrays indicated in the comments cards of program EXP.

The graphs are produced by two FORTRAN subroutines developed by the NPS computer staff.

Data checks are made on input:

Input check no. 1. Check if number of inputs is less than 'limit'.

Input check no. 2. Check if expected number of errors is nonnegative.

The listings and sample input and output are attached.

```
C     ************************************************************  TRAV0010
C     *                                                          *  TRAV0020
C     *  TRAVERSAL   PROGRAM TO COMPUTE THE TRAVERSAL            *  TRAV0030
C     *  PROBABILITIES OF A DIRECTED GRAPH FOR INPUTS BEGINNING  *  TRAV0040
C     *  AT THE START NODE AND ENDING AT THE EXIT NODE.          *  TRAV0050
C     *                                                          *  TRAV0060
C     *  VERSION 1.0  JUNE 1975.                                 *  TRAV0070
C     *                                                          *  TRAV0080
C     *  GORDON F. BRADLEY                                       *  TRAV0090
C     *  OPERATIONS RESEARCH DEPARTMENT                          *  TRAV0100
C     *  NAVAL POSTGRADUATE SCHOOL                               *  TRAV0110
C     *  MONTEREY ,CALIFORNIA 94340                              *  TRAV0120
C     *  PHONE 408-646-2769     AUTOVON  479-2769                *  TRAV0130
C     *  PHONE 408-646-2471     AUTOVON  479-2471                *  TRAV0140
C     *                                                          *  TRAV0150
C     *  INPUT.                                                  *  TRAV0160
C     *  CARD 1. (20A4) ALPHANUMERIC TITLE CARD.                 *  TRAV0170
C     *  CARD 2. (1X,I2,1X,I2) START NODE NUMBER AND EXIT NODE   *  TRAV0180
C     *       NUMBER                                             *  TRAV0190
C     *  CARD 3 ETC.   (1X,I2,1X,I2,1X,A1,2X,F10.5) FOR EACH ARC OF *  TRAV0200
C     *       THE GRAPH,TAIL NODE NUMBER,HEAD NODE NUMBER,       *  TRAV0210
C     *       ALPHA CHARACTER 'R' INDICATES THAT THE ARC MAY BE  *  TRAV0220
C     *       REPEATED,BRANCH PROBABILITY WHICH IS THE PROB.     *  TRAV0230
C     *       THAT THE ARC WILL BE TRAVERSED IF YOU ARE AT ITS   *  TRAV0240
C     *       TAIL NODE.                                         *  TRAV0250
C     *  LAST CARD. 00 OR BLANKS IN COLUMNS 2 AND 3.             *  TRAV0260
C     *                                                          *  TRAV0270
C     *  ANY ARC THAT MAY BE TRAVERSED MORE THAN ONCE BY ANY PATH *  TRAV0280
C     *       FROM THE START NODE TO THE EXIT NODE SHOULD BE MARKED *  TRAV0290
C     *       WITH AN 'R' IN COLUMN 8 TO ASSURE THE CORRECT TRAVERSAL* TRAV0300
C     *       PROBABILITY.                                       *  TRAV0310
C     *                                                          *  TRAV0320
C     *  PUNCH OUTPUT FOR EXPECTED ERRORS PROGRAM.               *  TRAV0330
C     *  CARD 1.  TITLE CARD                                     *  TRAV0340
C     *  CARD 2.  CARD TO INSERT NO. OF INPUTS FOR EXPECTED ERRORS *  TRAV0350
C     *       PROGRAM.                                           *  TRAV0360
C     *  CARD 3 ETC. SAME AS INPUT EXCEPT FOR TRAVERSAL          *  TRAV0370
C     *       PROBABILITIES ADDED  (F10.5)  IN COLUMNS 21 TO 30. *  TRAV0380
C     *  LAST CARD.  00 IN COLUMNS 2 AND 3.                      *  TRAV0390
```

```
C      *                                                               *   TRAV04C0
C      *  NCTE.  IDIM+1 IS DIMENSION OF ALL ARRAYS IN THIS PROGRAM     *   TRAV0410
C      *       AND ITS SUBROUTINES ,EXCEPT TITLE WHICH IS FIXED AT 20.*   TRAV0420
C      *                                                               *   TRAV0430
C      *  IF SPACE IS CRITICAL, IT WOULD BE POSSIBLE TO REWRITE THE    *   TRAV0440
C      *       PROGRAM TO REMOVE THE MATRIX P( , ) BY PUTTING THE      *   TRAV0450
C      *       PHONY ARC FIRST RATHER THAN LAST.                       *   TRAV0460
C      *                                                               *   TRAV0470
C      *  INPUT CHECK NO. 1.  INPUT FORMAT FORCES NODE NUMBERS TO BE   *   TRAV0480
C      *       LESS THAN 100.                                          *   TRAV0490
C      *  INPUT CHECK NO. 2.  CHECK FOR INPUT LIMIT OF 'IDIM' ARCS.    *   TRAV0500
C      *  INPUT CHECK NO. 3.  EXIT NODE SHOULD HAVE NO ARCS BRANCHING  *   TRAV0510
C      *       AWAY FROM IT.                                           *   TRAV0520
C      *  INPUT CHECK NO. 4.  NO BRANCH PROBABILITY SHOULD BE LESS     *   TRAV0530
C      *       THAN ZERO OR GREATER THAN ONE.                          *   TRAV0540
C      *  INPUT CHECK NO. 5.  CHECK IF THERE EXIST PATHS FROM          *   TRAV0550
C      *       START NODE THAT DO  NOT REACH EXIT NODE.                *   TRAV0560
C      *  INPUT CHECK NO. 6.  CHECK THAT SUM OF BRANCH PROBABILITIES   *   TRAV0570
C      *       FROM ALL NODES EQUAL APPROXIMATELY                      *   TRAV0580
C      *       ONE. IF IT DOES NOT, PRINT WARNING AND THEN DIVIDE      *   TRAV0590
C      *       BY SUM TO MAKE IT EQUAL TO ONE.                         *   TRAV0600
C      *  INPUT CHECK NO. 7.  CHECK THAT START NODE IS THE TAIL        *   TRAV0610
C      *       OF SOME ARC AND THAT EXIT NODE IS THE HEAD OF SOME      *   TRAV0620
C      *       ARC.                                                    *   TRAV0630
C      *                                                               *   TRAV0640
C      *  ASSUMES SUBROUTINE MARKOV WHICH IN TURN ASSUMES SUBROUTINE   *   TRAV0650
C      *       ISOLVE.                                                 *   TRAV0660
C      *                                                               *   TRAV0670
C      ***************************************************************   TRAV0680
C                                                                          TRAV0690
C                                                                          TRAV0700
      COMMON P(100,100),T(100,100),X(100,100),IST(100)
      DIMENSION ITAIL(100),IHEAD(100),REP(100),BRAN(100),TRAV(100),        TRAV0720
     1 Z(100),SAVE(100),TITLE(20)                                          TRAV0730
```

```
      DATA RLST/1HR/                                                    TRAV0740
      INTEGER CREER                                                     TRAV0750
C                                                                       TRAV0760
C IDIM MUST BE SET EQUAL TO DIMENSION OF P,T,X,IST     MINUS ONE.       TRAV0770
      IDIM = 99                                                         TRAV0780
C                                                                       TRAV0790
C                                                                       TRAV0800
C READ AND PRINT TITLE OF GRAPH.                                        TRAV0810
C                                                                       TRAV0820
    1 WRITE(6,860)                                                      TRAV0830
      READ(5,310)(TITLE(I),I=1,20)                                      TRAV0840
      WRITE(6,820) (TITLE(I),I=1,20)                                    TRAV0850
C                                                                       TRAV0860
C READ AND PRINT NUMBER OF START AND EXIT NODES.                        TRAV0870
C                                                                       TRAV0880
      READ(5,330) ISTART,IEXIT                                          TRAV0890
      WRITE(6,840) ISTART,IEXIT                                         TRAV0900
C                                                                       TRAV0910
C READ IN AND COUNT ARCS                                                TRAV0920
C                                                                       TRAV0930
      N=0                                                               TRAV0940
   20 N = N + 1                                                         TRAV0950
C     *  INPUT CHECK NO. 2.  CHECK FOR INPUT LIMIT OF 'IDIM' ARCS.      TRAV0960
      IF(N.LE.IDIM+1) GO TO 30                                          TRAV0970
      WRITE(6,930) IDIM                                                 TRAV0980
      STOP                                                              TRAV0990
   30 READ(5,850) ITAIL(N),IHEAD(N),REP(N), BRAN(N)                     TRAV1000
      IF(ITAIL(N).NE.0) GO TO 20                                        TRAV1010
      N = N - 1                                                         TRAV1020
C                                                                       TRAV1030
C     *  INPUT CHECK NO. 4.  NO BRANCH PROBABILITY SHOULD BE LESS       TRAV1040
C     *       THAN ZERO OR GREATER THAN ONE.                            TRAV1050
C     *  INPUT CHECK NO. 3.  EXIT NODE SHOULD HAVE NO ARCS BRANCHING    TRAV1060
C     *       AWAY FROM IT.                                             TRAV1070
```

```
C      *                                                               *   TRAV04C0
C      *  NCTE.  IDIM+1 IS DIMENSION OF ALL ARRAYS IN THIS PROGRAM     *   TRAV0410
C      *       AND ITS SUBROUTINES ,EXCEPT TITLE WHICH IS FIXED AT 20.*   TRAV0420
C      *                                                               *   TRAV0430
C      *  IF SPACE IS CRITICAL, IT WOULD BE POSSIBLE TO REWRITE THE    *   TRAV0440
C      *       PROGRAM TO REMOVE THE MATRIX P( , ) BY PUTTING THE      *   TRAV0450
C      *       PHONY ARC FIRST RATHER THAN LAST.                       *   TRAV0460
C      *                                                               *   TRAV0470
C      *  INPUT CHECK NO. 1.  INPUT FORMAT FORCES NODE NUMBERS TO BE   *   TRAV0480
C      *       LESS THAN 100.                                          *   TRAV0490
C      *  INPUT CHECK NO. 2.  CHECK FOR INPUT LIMIT OF 'IDIM' ARCS.    *   TRAV0500
C      *  INPUT CHECK NO. 3.  EXIT NODE SHOULD HAVE NO ARCS BRANCHING  *   TRAV0510
C      *       AWAY FROM IT.                                           *   TRAV0520
C      *  INPUT CHECK NO. 4.  NO BRANCH PROBABILITY SHOULD BE LESS     *   TRAV0530
C      *       THAN ZERO OR GREATER THAN ONE.                          *   TRAV0540
C      *  INPUT CHECK NO. 5.  CHECK IF THERE EXIST PATHS FROM          *   TRAV0550
C      *       START NODE THAT DO  NOT REACH EXIT NODE.                *   TRAV0560
C      *  INPUT CHECK NO. 6.  CHECK THAT SUM OF BRANCH PROBABILITIES   *   TRAV0570
C      *       FROM ALL NODES EQUAL APPROXIMATELY                      *   TRAV0580
C      *       ONE. IF IT DOES NOT, PRINT WARNING AND THEN DIVIDE      *   TRAV0590
C      *       BY SUM TO MAKE IT EQUAL TO ONE.                         *   TRAV0600
C      *  INPUT CHECK NO. 7.  CHECK THAT START NODE IS THE TAIL        *   TRAV0610
C      *       OF SOME ARC AND THAT EXIT NODE IS THE HEAD OF SOME      *   TRAV0620
C      *       ARC.                                                    *   TRAV0630
C      *                                                               *   TRAV0640
C      *  ASSUMES SUBROUTINE MARKOV WHICH IN TURN ASSUMES SUBROUTINE   *   TRAV0650
C      *       ISOLVE.                                                 *   TRAV0660
C      *                                                               *   TRAV0670
C      *****************************************************************   TRAV0680
C                                                                          TRAV0690
                                                                           TRAV0700
      COMMON P(100,100),T(100,100),X(100,100),IST(100)
      DIMENSION ITAIL(100),IHEAD(100),REP(100),BRAN(100),TRAV(100),        TRAV0720
     1 Z(100),SAVE(100),TITLE(20)                                          TRAV0730
```

```
      DATA BLET/1HR/                                                    TRAV0740
      INTEGER CREEP                                                     TRAV0750
C                                                                       TRAV0760
C  IDIM MUST BE SET EQUAL TO DIMENSION OF P,T,X,IST     MINUS ONE.      TRAV0770
      IDIM = 99                                                         TRAV0780
C                                                                       TRAV0790
C                                                                       TRAV0800
C  READ AND PRINT TITLE OF GRAPH.                                       TRAV0810
C                                                                       TRAV0820
    1 WRITE(6,860)                                                      TRAV0830
      READ(5,310)(TITLE(I),I=1,20)                                      TRAV0840
      WRITE(6,820) (TITLE(I),I=1,20)                                    TRAV0850
C                                                                       TRAV0860
C  READ AND PRINT NUMBER OF START AND EXIT NODES.                       TRAV0870
C                                                                       TRAV0880
      READ(5,330) ISTART,IEXIT                                          TRAV0890
      WRITE(6,840) ISTART,IEXIT                                         TRAV0900
C                                                                       TRAV0910
C  READ IN AND COUNT ARCS                                               TRAV0920
C                                                                       TRAV0930
      N=0                                                               TRAV0940
   20 N = N + 1                                                         TRAV0950
C     *  INPUT CHECK NO. 2.  CHECK FOR INPUT LIMIT OF 'IDIM' ARCS.      TRAV0960
      IF(N.LE.IDIM+1) GO TO 30                                          TRAV0970
      WRITE(6,930) IDIM                                                 TRAV0980
      STOP                                                              TRAV0990
   30 READ(5,850) ITAIL(N),IHEAD(N),REP(N), BRAN(N)                     TRAV1000
      IF(ITAIL(N).NE.0) GO TO 20                                        TRAV1010
      N = N - 1                                                         TRAV1020
C                                                                       TRAV1030
C     *  INPUT CHECK NO. 4.  NO BRANCH PROBABILITY SHOULD BE LESS       TRAV1040
C     *        THAN ZERO OR GREATER THAN ONE.                           TRAV1050
C     *  INPUT CHECK NO. 3.  EXIT NODE SHOULD HAVE NO ARCS BRANCHING    TRAV1060
C     *        AWAY FROM IT.                                            TRAV1070
```

```
C                                                                       TRAV1080
      DO 50 I = 1,N                                                     TRAV1090
      IF(BRAN(I).GE.0.0.AND.BRAN(I).LE.1.0)  GO TO 40                   TRAV1100
      WRITE(6,940) I,BRAN(I)                                            TRAV1110
      STOP                                                              TRAV1120
   40 IF(ITAIL(I).EQ.IEXIT) WRITE(6,950 ) I                             TRAV1130
   50 CONTINUE                                                          TRAV1140
C                                                                       TRAV1150
C     *  INPUT CHECK NO. 7.  CHECK THAT START NODE IS THE TAIL          TRAV1160
C     *        OF SOME ARC AND THAT EXIT NODE IS THE HEAD OF SOME       TRAV1170
C     *        ARC.                                                     TRAV1180
C                                                                       TRAV1190
      DO 60 I = 1,N                                                     TRAV1200
      IF(IHEAD(I).EQ.IEXIT) GO TO 70                                    TRAV1210
   60 CONTINUE                                                          TRAV1220
      WRITE(6,970)                                                      TRAV1230
      STOP                                                              TRAV1240
   70 DO 80 I = 1,N                                                     TRAV1250
      IF(ITAIL(I).EQ.ISTART) GO TO 90                                   TRAV1260
   80 CONTINUE                                                          TRAV1270
      WRITE(6,980)                                                      TRAV1280
      STOP                                                              TRAV1290
C                                                                       TRAV1300
C PRINT NUMBER OF ARCS.                                                 TRAV1310
C                                                                       TRAV1320
   90 WRITE(6,860) N                                                    TRAV1330
      ORDER = N + 1                                                     TRAV1340
C                                                                       TRAV1350
C INITIALIZE PHONY  FINAL STATE.                                        TRAV1360
C                                                                       TRAV1370
      ITAIL(ORDER) = IEXIT                                              TRAV1380
      IHEAD(ORDER) = 9999                                               TRAV1390
      BRAN(ORDER) = 1.0                                                 TRAV1400
C                                                                       TRAV1410
C SET UP TRANSITION MATRIX.                                             TRAV1420
C                                                                       TRAV1430
      DO 130 I = 1, N                                                   TRAV1440
      SUM = 0.0                                                         TRAV1450
      DO 100 J = 1, ORDER                                               TRAV1460
```

```
      P(I,J) = 0.0                                                      TRAV1470
      IF(IHEAD(I).NE.ITAIL(J)) GO TO 100                                TRAV1480
      P(I,J) = BRAN(J)                                                  TRAV1490
      SUM = SUM + BRAN(J)                                               TRAV1500
  100 CONTINUE                                                          TRAV1510
C     *   INPUT CHECK NO. 6.  CHECK THAT SUM OF BRANCH PROBABILITIES    TRAV1520
C     *         FROM ALL NODES EQUAL APPROXIMATELY                      TRAV1530
C     *         ONE. IF IT DOES NOT, PRINT WARNING AND THEN DIVIDE      TRAV1540
C     *         BY SUM TO MAKE IT EQUAL TO ONE.                         TRAV1550
      IF(SUM.GT.1.005.OR.SUM.LT.0.995)  WRITE(6,900)IHEAD(I),SUM        TRAV1560
C DIVIDE THROUGH BY SUM SO THAT ROW SUM IS 1.0                          TRAV1570
  110 DO 120 J = 1,ORDER                                                TRAV1580
  120 P(I,J) = P(I,J)/SUM                                               TRAV1590
  130 CONTINUE                                                          TRAV1600
C                                                                       TRAV1610
C SET UP ROW FOR PHONY  FINAL STATE.                                    TRAV1620
C                                                                       TRAV1630
      DO 140 J = 1,ORDER                                                TRAV1640
  140 P(ORDER,J) = 0.0                                                  TRAV1650
      P(ORDER,ORDER) = 1.0                                              TRAV1660
C                                                                       TRAV1670
C CALL MARKOV SUBROUTINE.                                               TRAV1680
C                                                                       TRAV1690
      CALL MARKOV(ORDER)                                                TRAV1700
C                                                                       TRAV1710
C COMPUTE INITIAL PROBABILITIES OF BEGINNING ARCS.                      TRAV1720
C                                                                       TRAV1730
      SUM = 0.0                                                         TRAV1740
      DO 150 I = 1,ORDER                                                TRAV1750
      Z(I) = 0.0                                                        TRAV1760
      IF(ITAIL(I).EQ.ISTART)  Z(I) = BRAN(I)                            TRAV1770
  150 SUM = SUM + Z(I)                                                  TRAV1780
C                                                                       TRAV1790
C     *   INPUT CHECK NO. 6.  CHECK THAT SUM OF BRANCH PROBABILITIES    TRAV1800
C     *         FROM ALL NODES EQUAL APPROXIMATELY                      TRAV1810
C     *         ONE. IF IT DOES NOT, PRINT WARNING AND THEN DIVIDE      TRAV1820
C     *         BY SUM TO MAKE IT EQUAL TO ONE.                         TRAV1830
```

```
C                                                                       TRAV1840
      IF(SUM.GT.1.005.OR.SUM.LT.0.995)  WRITE(6,900)ISTART,SUM          TRAV1850
C DIVIDE THROUGH BY SUM SO THAT ROW SUM IS 1.0                          TRAV1860
      DO 160 I = 1,ORDER                                                TRAV1870
  160 Z(I) = Z(I)/SUM                                                   TRAV1880
C FOR DEBUG PURPOSES, PRINT Z(I)                                        TRAV1890
C     WRITE(6,998) (Z(I), I = 1,ORDER)                                  TRAV1900
C 998 FORMAT( 5H Z(I), 10F10.7,/,(5X,10F10.7))                          TRAV1910
C                                                                       TRAV1920
C COMPUTE TRAVERSAL PROBABILITIES.                                      TRAV1930
C                                                                       TRAV1940
      DO 300 J = 1,N                                                    TRAV1950
  300 TRAV(J) = 0.0                                                     TRAV1960
      DO 320 I = 1,N                                                    TRAV1970
      IF(Z(I).EQ.0.0) GO TO 320                                         TRAV1980
      DO 310 J = 1,N                                                    TRAV1990
  310 TRAV(J) = TRAV(J) + Z(I)*X(I,J)                                   TRAV2000
  320 CONTINUE                                                          TRAV2010
C                                                                       TRAV2020
C REDO ANALYSIS FOR ARCS MARKED 'R' FOR REPEAT.                         TRAV2030
C                                                                       TRAV2040
      DO 440 I = 1, N                                                   TRAV2050
      IF(REP(I).NE.RLET) GO TO 440                                      TRAV2060
C SAVE I ROW OF P(I,J) AND SUBSTITUTE DIRECT LINK TO PHONY ARC.         TRAV2070
      DO 410 J = 1,ORDER                                                TRAV2080
      SAVE(J) = P(I,J)                                                  TRAV2090
  410 P(I,J) = 0.0                                                      TRAV2100
      P(I,ORDER) = 1.0                                                  TRAV2110
      CALL MARKOV(ORDER)                                                TRAV2120
C COMPUTE NEW TRAVERSAL PROB. FOR ARC I.                                TRAV2130
      TRAV(I) = 0.0                                                     TRAV2140
      DO 420 K = 1,N                                                    TRAV2150
      IF(Z(K).EQ.0.0)  GO TO 420                                        TRAV2160
      TRAV(I) = TRAV(I) + Z(K)*X(K,I)                                   TRAV2170
  420 CONTINUE                                                          TRAV2180
C RESTORE ROW I OF P(I,J)                                               TRAV2190
```

```
      DO 430 J = 1,ORDER                                                TRAV2200
  430 P(I,J) = SAVE(J)                                                  TRAV2210
  440 CONTINUE                                                          TRAV2220
C                                                                       TRAV2230
C     *  INPUT CHECK NO. 5.  CHECK IF THERE EXIST PATHS FROM            TRAV2240
C     *        START NODE THAT DO   NOT REACH EXIT NODE.                TRAV2250
C                                                                       TRAV2260
      SUM = 0.0                                                         TRAV2270
      DO 600 I = 1,N                                                    TRAV2280
      IF(IHEAD(I).EQ.IEXIT) SUM = SUM + TRAV(I)                         TRAV2290
  600 CONTINUE                                                          TRAV2300
      IF(SUM.LT.0.995)  WRITE(6,960) SUM                                TRAV2310
C                                                                       TRAV2320
C PRINT OUT INPUT DATA AND TRAVERSAL PROBABILITY. PUNCH ALSO.           TRAV2330
C                                                                       TRAV2340
      WRITE(6,870)                                                      TRAV2350
       PUNCH 810,  (TITLE(I),I=1,20)                                    TRAV2360
      PUNCH 910                                                         TRAV2370
      DO 500  I=1,N                                                     TRAV2380
      PUNCH 950,ITAIL(I),IHEAD(I),REP(I),BRAN(I),TRAV(I)                TRAV2390
  500 WRITE(6,880) ITAIL(I),IHEAD(I),REP(I),BRAN(I),TRAV(I)             TRAV2400
      WRITE(6,890)                                                      TRAV2410
      PUNCH 920                                                         TRAV2420
      STOP                                                              TRAV2430
C                                                                       TRAV2440
C                                                                       TRAV2450
  800 FORMAT(70H1ARC TRAVERSAL PROBABILITIES FOR ERROR DETECTION        TRAV2460
     1VERSION 1.0     )                                                 TRAV2470
  810 FORMAT(20A4)                                                      TRAV2480
  820 FORMAT(//,   17H PROGRAM TITLE:   ,  20A4    )                    TRAV2490
  830 FORMAT(  1X,   I2,    1X,    I2    )                              TRAV2500
  840 FORMAT(//,  13H START NODE = ,I3,3X,12H EXIT NODE = ,  I3   )     TRAV2510
  850 FORMAT(1X,   I2,   1X,   I2,    1X,   A1,   2X,   F10.5,   F10.5  )  TRAV2520
  860 FORMAT(//,   18H NUMBER OF ARCS = ,  I3 )                         TRAV2530
  870 FORMAT(//,     10X,   3HARC,   20X,     6HBRANCH,     8X,        TRAV2540
     110H TRAVERSAL,   /,   5X,    4HTAIL,    5X,  4HHEAD,  5X,    6HREPEAT,  TRAV2550
     2   5X,  6HPROB. ,    10X,     5HPROB.    )                       TRAV2560
  880 FORMAT(    5X,    I3,    6X,     I3,      8X,   A1,    3X,       F10.4,  TRAV2570
     1         6X,   F10.4          )                                  TRAV2580
  890 FORMAT(1H1)                                                       TRAV2590
  900 FORMAT(   5H NODE, I5, 21H SUM OF BRANCH PROB = , F10.5   ,       TRAV2600
     1   21H POSSIBLE INPUT ERROR       )                              TRAV2610
  910 FORMAT(9H    INPUTS)                                              TRAV2620
  920 FORMAT(3H000)                                                     TRAV2630
  930 FORMAT( 32H EXCEED ARC LIMIT WHICH EQUALS    , I5   )            TRAV2640
  940 FORMAT(20H BRANCH PROB. OF ARC,   I5, 3H IS, F10.5    )           TRAV2650
  950 FORMAT(12H WARNING.ARC ,  I5,  25H BRANCHES FROM EXIT NODE.     )  TRAV2660
  960 FORMAT( 58H WARNING, SUM OF TRAVERSAL PROBABILITIES INTO EXIT NODETRAV2670
     1 IS    ,   F10.5,   /,   78H MAY INDICATE THERE IS A PATH FROM START NOTRAV2680
     2E THAT DOES NOT REACH EXIT NODE.          )                      TRAV2690
  970 FORMAT(  38H EXIT NODE IS NOT THE HEAD OF ANY ARC.    )           TRAV2700
  980 FORMAT(   4CH START NODE IS NOT THE TAIL OF ANY ARC.     )        TRAV2710
      END                                                               TRAV2720
```

```
      SUBROUTINE  MARKOV(ORDER)                                          MARK0010
C  ORIGINALLY WRITTEN FOR CDC 6600, CONVERTED TO 360/67 4 NOV 74          MARK0020
C    NO GUARANTEES OF PERFORMANCE - G. BROWN, NPS, 93940                 MARK0030
      COMMON P(100,100),T(100,100),X(100,100),IST(100)                   MARK0040
      DIMENSION IB(100)                                                  MARK0050
      INTEGER ORDER                                                      MARK0060
C                                                                        MARK0070
C IDIM MUST BE SET EQUAL TO DIMENSION OF P,T,X,IST,IB                    MARK0080
      IDIM = 100                                                         MARK0090
C                                                                        MARK0100
      DET = 0.0                                                          MARK0110
C IDENTIFY IDENTITY PART OF MATRIX                                       MARK0120
      NB = 1                                                             MARK0130
      DO 100 I=1,ORDER                                                   MARK0140
      IF ( P(I,I) .NE. 1.0 ) GO TO 100                                   MARK0150
      IB(NB) = I                                                         MARK0160
      NB = NB + 1                                                        MARK0170
  100 CONTINUE                                                           MARK0180
      K = NB - 1                                                         MARK0190
C INDICATE REORDERING FOR NON IDENTITY PART.                             MARK0200
      DO 110 I = 1,ORDER                                                 MARK0210
      DO 105 J=1,K                                                       MARK0220
      IF ( IB(J) .EQ. I ) GO TO 110                                      MARK0230
  105 CONTINUE                                                           MARK0240
      IB(NB) = I                                                         MARK0250
      NB = NB + 1                                                        MARK0260
  110 CONTINUE                                                           MARK0270
C MOVE P(I,J) TO T(I,J) REORDERING ROWS AND COLUMNS.                     MARK0280
      DO 120 I=1,ORDER                                                   MARK0290
      II = IB(I)                                                         MARK0300
      DO 120 J=1,ORDER                                                   MARK0310
      JJ = IB(J)                                                         MARK0320
      T(J,I) = P(II,JJ)                                                  MARK0330
  120 CONTINUE                                                           MARK0340
```

```
      L = K + 1                                                          MARK0350
      DO 200 I=L,ORDER                                                   MARK0360
      IK = I-K                                                           MARK0370
      II = IB(I)                                                         MARK0380
      DO 190 J=L,ORDER                                                   MARK0390
      JK = J - K                                                         MARK0400
      JJ = IB(J)                                                         MARK0410
      T(IK,JK) = -P(II,JJ)                                               MARK0420
  190 CONTINUE                                                           MARK0430
      T(IK,IK) = T(IK,IK) + 1.0                                          MARK0440
  200 CONTINUE                                                           MARK0450
C CALL INVERT FOR NON IDENTITY PART OF MATRIX.                           MARK0460
      M = ORDER - K                                                      MARK0470
      I = ISCLVE(ICIM,M,-M,T,X,DET,IST)                                  MARK0480
C IF MATRIX IS NONSINGULAR, PRINT MESSAGE.                               MARK0490
      IF(I.EQ.1) WRITE(6,834)                                            MARK0500
  834 FORMAT(30H1MATRIX SINGULAR, CHECK INPUT.    )                      MARK0510
C FOR DEBUG PURPOSES, PRINT ORIGINAL AND NONIDENTITY INVERSE.            MARK0520
C     PRINT 80                                                           MARK0530
C  80 FORMAT(    28H1ORIGINAL TRANSITION MATRIX.    )                   MARK0540
C     DO 90  I = 1,ORDER                                                 MARK0550
C     PRINT 141,  I                                                      MARK0560
C  90 PRINT 81, (P(I,J),J=1,ORDER )                                      MARK0570
C     PRINT 32, (IB(J),J=L,ORDER )                                       MARK0580
C     DO 220 I=1,M                                                       MARK0590
C     J = I + K                                                          MARK0600
C     PRINT 141, IB(J)                                                   MARK0610
C     PRINT 81, (X(I,J),J=1,M )                                          MARK0620
C 220 CONTINUE                                                           MARK0630
C  32 FORMAT( 54H1INVERSE OF NONIDENTITY PART WITH CORRESPONDING STATES  MARK0640
C    1    ,/,  (2X,10I10))                                             MARK0650
C 141 FORMAT(1H ,I3, 1H-  )                                              MARK0660
C  81 FORMAT(      5X, 10F10.7, /, (5X, 10F10.7) )                       MARK0670
      RETURN                                                             MARK0680
      END                                                                MARK0690
```

```
      FUNCTION ISOLVE (IDSM, NE, NC, A, B, DET, IC)                     ISOL0010
C     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ISOL0020
C     **************************************************************** ISOL0030
C     *                                                              * ISOL0040
C     *  SUBROUTINE TO SOLVE SIMULTANEOUS LINEAR EQUATIONS           * ISOL0050
C     *                                                              * ISOL0060
C     *  ARGUMENTS -                                                 * ISOL0070
C     *    IDSM     DIMENSIONED SIZE OF COEFFICIENT MATRIX          * ISOL0080
C     *    NE       ACTUAL NUMBER OF EQUATIONS FOR THIS CALL        * ISOL0090
C     *    NC       NUMBER OF COLUMNS IN CONSTANT MATRIX            * ISOL0100
C     *    A        COEFFICIENT MATRIX                              * ISOL0110
C     *    B        CONSTANT MATRIX                                 * ISOL0120
C     *    DET      DETERMINANT VALUE OF COEFFICIENT MATRIX         * ISOL0130
C     *    IC       TEMPORARY STORAGE FOR SUBROUTINE                * ISOL0140
C     *    ISOLVE   RETURNS 0 IF GOOD, 1 IF SINGULAR                * ISOL0150
C     *                                                             * ISOL0160
C     *  NOTE -     IF NE IS NEGATIVE, ONLY THE DETERMINANT VALUE IS * ISOL0170
C     *             REQUIRED AS OUTPUT, MATRIX B AND ARRAY IC ARE   * ISOL0180
C     *             NOT USED.                                       * ISOL0190
C     *             IF NC IS NEGATIVE, THE INVERSE OF THE           * ISOL0200
C     *             COEFFICIENT MATRIX IS REQUIRED, MATRIX B IS SET * ISOL0210
C     *             UP AS THE IDENTITY MATRIX                       * ISOL0220
C     *                                                             * ISOL0230
C     **************************************************************** ISOL0240
C                                                                       ISOL0250
      DIMENSION A(1), B(1), IC(1)                                       ISOL0260
C     INITIALIZE                                                        ISOL0270
      N = IABS(NE)                                                      ISOL0280
      M = IABS(NC)                                                      ISOL0290
      ID = IDSM                                                         ISOL0300
      ISOLVE = 0                                                        ISOL0310
      DET = 1.                                                          ISOL0320
      IF (NE) 5000, 2000, 5100                                          ISOL0330
 5000 IDV = 1                                                           ISOL0340
      GO TO 15                                                          ISOL0350
 5100 IDV = 0                                                           ISOL0360
```

```
      DO 1 I=1,N                                                        ISOL0370
    1 IC(I) = I                                                         ISOL0380
      IF (NC) 5, 2000, 15                                               ISOL0390
C     INVERSE REQUIRED                                                  ISOL0400
    5 ISUB2 = 0                                                         ISOL0410
      DO 10 J=1,N                                                       ISOL0420
      ISUB1 = ISUB2                                                     ISOL0430
      DO 6 I=1,N                                                        ISOL0440
      ISUB1 = ISUB1 + 1                                                 ISOL0450
    6 B(ISUB1) = 0.                                                     ISOL0460
      ISUB1 = ISUB2 + J                                                 ISOL0470
      B(ISUB1) = 1.                                                     ISOL0480
   10 ISUB2 = ISUB2 + ID                                                ISOL0490
C     START MAIN LOOP                                                   ISOL0500
   15 DO 1000 L=1,N                                                     ISOL0510
      LM1 = L - 1                                                       ISOL0520
      DO 40 I=L,N                                                       ISOL0530
      PIVOT = 0.                                                        ISOL0540
      ISUB1 = ID * LM1 + I                                              ISOL0550
      ISUB2 = ISUB1                                                     ISOL0560
      DO 20 J=L,N                                                       ISOL0570
      IF (ABS(PIVOT) - ABS(A(ISUB1))) 17, 20, 20                        ISOL0580
   17 PIVOT = A(ISUB1)                                                  ISOL0590
      JB = J                                                            ISOL0600
   20 ISUB1 = ISUB1 + ID                                                ISOL0610
C     COMPUTE DETERMINANT                                               ISOL0620
      DET = DET * PIVOT                                                 ISOL0630
C     TEST FOR SINGULAR MATRIX                                          ISOL0640
      IF (PIVOT) 24, 2000, 24                                           ISOL0650
   24 DO 25 J=L,N                                                       ISOL0660
      A(ISUB2) = A(ISUB2) / PIVOT                                       ISOL0670
   25 ISUB2 = ISUB2 + ID                                                ISOL0680
      IF (ICVO) 29, 29, 35                                              ISOL0690
   29 ISUB1 = I                                                         ISOL0700
      DO 30 J=1,M                                                       ISOL0710
      B(ISUB1) = B(ISUB1) / PIVOT                                       ISOL0720
   30 ISUB1 = ISUB1 + ID                                                ISOL0730
```

```
   35 IF (I-L) 40, 39, 40                                                ISOLC740
   39 JP = JR                                                            ISOLC750
   40 CONTINUE                                                           ISOLC760
C     INTERCHANGE COLUMNS                                                ISOLC770
  100 IF (JP-L) 101, 260, 101                                           ISOLC780
  101 IF (IDV) 102, 102, 110                                            ISOLC790
  102 IT = IC(L)                                                        ISOLC800
      IC(L) = IC(JP)                                                    ISOLC810
      IC(JP) = IT                                                       ISOLC820
  110 IR = ID * LM1                                                     ISOLC830
      IT = ID * JP - ID                                                 ISOLC840
      DO 120 I=1,N                                                      ISOLC850
      ISUB1 = IR + I                                                    ISOLC860
      ISUB2 = IT + I                                                    ISOLC870
      S = A(ISUB1)                                                      ISOLC880
      A(ISUB1) = A(ISUB2)                                               ISOLC890
  120 A(ISUB2) = S                                                      ISOLC900
      DET = -DET                                                        ISOLC910
C     REDUCE PIVOT COLUMN                                               ISOLC920
  260 IR = ID * LM1                                                     ISOLC930
      DO 400 I=1,N                                                      ISOLC940
      IR = IR + 1                                                       ISOLC950
      PIVOT = A(IR)                                                     ISOLC960
      IF (I-L) 261, 400, 261                                            ISOLC970
  261 IF (PIVOT) 270, 400, 270                                          ISOLC980
  270 ISUB1 = L                                                         ISOLC990
      ISUB2 = I                                                         ISOL1000
      DO 360 J=1,N                                                      ISOL1010
      IF (J- L ) 300, 300, 280                                          ISOL1020
  280 S = PIVOT * A(ISUB1)                                              ISOL1030
      A(ISUB2) = A(ISUB2) - S                                           ISOL1040
      IF (ABS(A(ISUB2)) - ABS(3.E-10* S)) 290, 300, 300                 ISOL1050
  290 A(ISUB2) = 0.                                                     ISOL1060
  300 IF (IDV) 310, 310, 350                                            ISOL1070
  310 IF (J - M) 320, 320, 350                                          ISOL1080
  320 B(ISUB2) = B(ISUB2) - PIVOT * B(ISUB1)                            ISOL1090
  350 ISUB1 = ISUB1 + ID                                                ISOL1100
  360 ISUB2 = ISUB2 + ID                                                ISOL1110
  400 CONTINUE                                                          ISOL1120
 1000 CONTINUE                                                          ISOL1130
      IF (IDV) 1100, 1100, 1500                                         ISOL1140
C     REARRANGE VARIABLES                                               ISOL1150
 1100 DO 1200 L=1,N                                                     ISOL1160
      ISUB1 = IC(L)                                                     ISOL1170
      ISUB2 = L                                                         ISOL1180
      DO 1200 J=1,M                                                     ISOL1190
      A(ISUB1) = B(ISUB2)                                               ISOL1200
      ISUB1 = ISUB1 + ID                                                ISOL1210
 1200 ISUB2 = ISUB2 + ID                                                ISOL1220
 1500 RETURN                                                            ISOL1230
C     SINGULAR MATRIX                                                   ISOL1240
 2000 ISOLVE = 1                                                        ISOL1250
      GO TO 1500                                                        ISOL1260
      END                                                               ISOL1270
```

```
  ERROR DETECTION MODEL   FIGURE 1
S01E16
A06-12 R      1.0
A12-03 R       .5
A12-16         .5
A11-13        1.0
A13-16        1.0
A14-16        1.0
AC7-11         .5
A08-14        1.0
A01-02        1.0
A02-03         .5
AC2-04         .5
A04-07         .3333
AC4-08         .3333
A04-09         .3333
AC7-10         .5
AC9-14         .5
AC9-15         .5
A10-13        1.0
A15-16        1.0
AC3-05 R       .5
A03-06 R       .5
A05-12 R      1.0
00C
```

Card input to program TRAV

ARC TRAVERSAL PROBABILITIES FOR ERROR DETECTION VERSION 1.0

PROGRAM TITLE: ERROR DETECTION MODEL FIGURE 1

START NODE = 1 EXIT NODE = 16

NUMBER OF ARCS = 22

ARC TAIL	ARC HEAD	REPEAT	BRANCH PROB.	TRAVERSAL PROB.
6	12		1.0000	0.3333
12	3		0.5000	0.2500
11	13		1.0000	0.0833
13	16		1.0000	0.1667
14	16		1.0000	0.2500
7	11		0.5000	0.0833
8	14		1.0000	0.1667
1	2		1.0000	1.0000
2	3		0.5000	0.5000
12	16		0.5000	0.5000
2	4		0.5000	0.5000
4	7		0.3333	0.1667
4	8		0.3333	0.1667
4	9		0.3333	0.1667
7	10		0.5000	0.0833
9	14		0.5000	0.0833
9	15		0.5000	0.0833
10	13		1.0000	0.0833
15	16		1.0000	0.0833
3	5	R	0.5000	0.3333
3	6	R	0.5000	0.3333
5	12	R	1.0000	0.3333

Printed output of program TRAV

```
  ERROR DETECTION MODEL    FIGURE 1
    INPUTS
   6  12 R       1.00000    0.33333
  12   3 R       0.50000    0.25000
  12  16         0.50000    0.50000
  11  13         1.00000    0.08333
  13  16         1.00000    0.16667
  14  16         1.00000    0.25000
   7  11         0.50000    0.08333
   8  14         1.00000    0.16667
   1   2         1.00000    1.00000
   2   3         0.50000    0.50000
   2   4         0.50000    0.50000
   4   7         0.33330    0.16667
   4   8         0.33330    0.16667
   4   9         0.33330    0.16667
   7  10         0.50000    0.08333
   8  14         0.50000    0.08333
   9  15         0.50000    0.08333
  10  13         1.00000    0.08333
  15  16         1.00000    0.08333
   3   5 R       0.50000    0.33333
   3   6 R       0.50000    0.33333
   5  12 R       1.00000    0.33333
000
```

Card output from program TRAV

```
C     ************************************************************   EXP  0010
C     *                                                          *   EXP  0020
C     *  PROGRAM TO COMPUTE EXPECTED AND CUMMULATIVE NUMBER OF   *   EXP  0030
C     *  ERRORS DETECTED AND OUTPUT TABLE AND GRAPH OF THE RESULT.*   EXP  0040
C     *                                                          *   EXP  0050
C     *  VERSION 1.0  JUNE 1975.                                 *   EXP  0060
C     *                                                          *   EXP  0070
C     *  GORDON H. BRADLEY                                       *   EXP  0080
C     *  OPERATIONS RESEARCH DEPARTMENT                          *   EXP  0090
C     *  NAVAL POSTGRADUATE SCHOOL                               *   EXP  0100
C     *  MONTEREY ,CALIFORNIA 94340                              *   EXP  0110
C     *  PHONE 408-646-2769      AUTOVON  479-2769               *   EXP  0120
C     *  PHONE 408-646-2471      AUTOVON  479-2471               *   EXP  0130
C     *                                                          *   EXP  0140
C     *  INPUT.                                                  *   EXP  0150
C     *  CARD 1.  TITLE CARD OUTPUT FROM TRAVERSAL PROGRAM.      *   EXP  0160
C     *  CARD 2.  (I3) NUMBER OF INPUTS TO THE DIRECTED GRAPH.   *   EXP  0170
C     *  CARD 3 ETC.  TRAVERSAL PROB. OUTPUT FROM TRAVERSAL PROGRAM. *   EXP  0180
C     *  CARD N.  FINAL TRAVERSAL CARD,00 OR BLANK IN COLUMNS 2 AND 3*   EXP  0190
C     *       OUTPUT FROM TRAVERSAL PROGRAM.                     *   EXP  0200
C     *  CARD N+1 ETC.  (8F10.2)  EXPECTED NUMBER OF ERRORS FOR EACH *   EXP  0210
C     *       ARC. ORDERING SAME AS TRAVERSAL PROB. INPUT ABOVE. *   EXP  0220
C     *  LAST CARD.  NO FINAL CARD IS NECESSARY.                 *   EXP  0230
C     *                                                          *   EXP  0240
C     *  TO HANDLE TWO OR MORE GRAPHS BRANCH FROM END OF PROGRAM *   EXP  0250
C     *       'STOP' TO STATEMENT 1.                             *   EXP  0260
C     *  TO HANDLE TWO OR MORE EXPECTED ERROR DISTRIBUTIONS FOR A *   EXP  0270
C     *       GIVEN GRAPH, BRANCH FROM END OF PROGRAM 'STOP' TO  *   EXP  0280
C     *       STATEMENT 2.                                       *   EXP  0290
C     *                                                          *   EXP  0300
C     *  ASSUMES INPUT OF ARCS IS CORRECT SINCE PREVIOUS  PROGRAM *   EXP  0310
C     *       HAD REASONABLENESS CHECKS.                         *   EXP  0320
C     *  INPUT CHECK NO. 1.  CHECK IF NUMBER OF INPUTS IS LESS   *   EXP  0330
C     *       THAN 'LIMIT'.                                      *   EXP  0340
C     *  INPUT CHECK NO. 2.  CHECK IF EXPECTED NUMBER OF ERRORS  *   EXP  0350
C     *       IS NONNEGATIVE.                                    *   EXP  0360
C     *                                                          *   EXP  0370
C     *  NOTE.  'LIMIT' IS THE DIMENSION OF E,CUM AND X. DIMENSION *   EXP  0380
C     *       OF ITAIL,IHEAD,BRAN,TRAV, AND MU IS ASSUMED LARGE  *   EXP  0390
C     *       ENOUGH TO HANDLE ANY INPUT.                        *   EXP  0400
C     *                                                          *   EXP  0410
C     *  ASSUMES SUBROUTINE PLOTP WHICH IN TURN ASSUMES SUBROUTINE *   EXP  0420
C     *       UTPLOT.  THESE SUBROUTINES ARE USED FOR THE OUTPUT *   EXP  0430
C     *       OF THE GRAPHS ONLY.                                *   EXP  0440
C     *                                                          *   EXP  0450
C     ************************************************************   EXP  0460
C                                                                      EXP  0470
```

```
C                                                                       EXP  0480
      DIMENSION ITAIL(100),IHEAD(100),          BRAN(100),TRAV(100),    EXP  0490
     1  MU(100),TITLE(20),E(150),CUM(150),X(150)                        EXP  0500
      REAL MU                                                           EXP  0510
C                                                                       EXP  0520
C LIMIT MUST EQUAL DIMENSION OF E,CUM,X                                 EXP  0530
      LIMIT = 150                                                       EXP  0540
C                                                                       EXP  0550
C READ AND PRINT TITLE OF GRAPH.                                        EXP  0560
C                                                                       EXP  0570
    1 WRITE(6,800)                                                      EXP  0580
      READ(5,810)(TITLE(I),I=1,20)                                      EXP  0590
      WRITE(6,820) (TITLE(I),I=1,20)                                    EXP  0600
C                                                                       EXP  0610
C READ AND PRINT NUMBER OF INPUTS.                                      EXP  0620
C                                                                       EXP  0630
      READ(5,330) INPUTS                                                EXP  0640
C     *  INPUT CHECK NO. 1.  CHECK IF NUMBER OF INPUTS IS LESS          EXP  0650
C     *       THAN 'LIMIT'.                                            EXP  0660
      IF(INPUTS.LE.LIMIT) GO TO 10                                      EXP  0670
      WRITE(6,940) LIMIT                                                EXP  0680
      INPUTS = LIMIT                                                    EXP  0690
   10 WRITE(6,840) INPUTS                                               EXP  0700
C                                                                       EXP  0710
C READ IN AND COUNT NUMBER OF ARCS AND TRAVERSAL PROBABILITIES.         EXP  0720
C                                                                       EXP  0730
      N=0                                                               EXP  0740
   20 N = N + 1                                                         EXP  0750
      READ(5,850) ITAIL(N),IHEAD(N),          BRAN(N),TRAV(N)           EXP  0760
      IF(ITAIL(N).NE.0) GO TO 20                                        EXP  0770
      N = N - 1                                                         EXP  0780
      WRITE(6,860) N                                                    EXP  0790
C                                                                       EXP  0800
C READ IN EXPECTED NUMBER OF ERRORS FOR EACH ARC                        EXP  0810
C                                                                       EXP  0820
    2 READ(5,865)  ( MU(I),  I = 1,N   )                                EXP  0830
C                                                                       EXP  0840
C PRINT OUT INPUT DATA ,   TRAVERSAL PROBABILITY AND EXPECTED NUMBER    EXP  0850
C   OF ERRORS.                                                          EXP  0860
C                                                                       EXP  0870
      WRITE(6,870)                                                      EXP  0880
      TOTAL = 0.0                                                       EXP  0890
      DO  30  I=1,N                                                     EXP  0900
      TOTAL = TOTAL + MU(I)                                             EXP  0910
      WRITE(6,880) ITAIL(I),IHEAD(I),          BRAN(I),TRAV(I),MU(I)    EXP  0920
C     *  INPUT CHECK NO. 2.  CHECK IF EXPECTED NUMBER OF ERRORS         EXP  0930
C     *       IS NONNEGATIVE.                                          EXP  0940
      IF(MU(I).LT.0.0) WRITE(6,950)    MU(I)                            EXP  0950
   30 CONTINUE                                                          EXP  0960
C                                                                       EXP  0970
C OUTPUT TOTAL EXPECTED NUMBER OF ERRORS.                               EXP  0980
C                                                                       EXP  0990
      WRITE(6,890)   TOTAL                                              EXP  1000
```

```
C COMPUTE EXPECTED NUMBER AND CUMMULATIVE NUMBER OF EXPECTED ERRORS.      EXP  1030
C                                                                         EXP  1040
      CUM(1) = 0.0                                                        EXP  1050
      DO 60 I=1,INPUTS                                                    EXP  1060
      E(I)=0.0                                                            EXP  1070
      DO 50 J=1,N                                                         EXP  1080
      E(I) = E(I)   +  TRAV(J)*MU(J)                                      EXP  1090
   50 MU(J) = MU(J)*(1.0 - TRAV(J))                                       EXP  1100
   60 CUM(I+1)=CUM(I)+E(I)                                                EXP  1110
C                                                                         EXP  1120
C PRINT   EXPECTED NUMBER AND CUMMULATIVE NUMBER OF EXPECTED ERRORS.      EXP  1130
C                                                                         EXP  1140
      WRITE(6,900) (I,E(I),CUM(I+1),I=1,INPUTS    )                       EXP  1150
C                                                                         EXP  1160
C PRINT GRAPH OF EXPECTED NUMBER AND CUMMULATIVE NUMBER OF EXPECTED       EXP  1170
C  ERRORS.                                                                EXP  1180
C                                                                         EXP  1190
      WRITE(6,910)                                                        EXP  1200
      WRITE(6,320) (TITLE(I),I=1,20)                                      EXP  1210
      DO 500 I  = 1,INPUTS                                                EXP  1220
  500 X(I) = I                                                            EXP  1230
      CALL PLOTP(X,E,INPUTS,0)                                            EXP  1240
      WRITE(6,920)                                                        EXP  1250
      WRITE(6,930)                                                        EXP  1260
      WRITE(6,820) (TITLE(I),I=1,20)                                      EXP  1270
      CALL PLOTP(X,CUM,INPUTS,0)                                          EXP  1280
      WRITE(6,920)                                                        EXP  1290
      WRITE(6,960)                                                        EXP  1300
      STOP                                                                EXP  1310
C                                                                         EXP  1320
C                                                                         EXP  1330
  800 FORMAT( 52H1EXPECTED NUMBER OF ERRORS DETECTED     VERSION 1.0    ) EXP  1340
  810 FORMAT(20A4)                                                        EXP  1350
  820 FORMAT(//,   17H PROGRAM TITLE:  ,  20A4 , //   )                   EXP  1360
  830 FORMAT(  I3    )                                                    EXP  1370
  840 FORMAT(              9H INPUTS = , I3     )                         EXP  1380
  850 FORMAT(1X,   I2,  1X,  I2,   4X,          F10.5,  F10.5  )         EXP  1390
  860 FORMAT(/ ,   18H NUMBER OF ARCS = ,  I3 )                           EXP  1400
  865 FORMAT(     8F10.2    )                                             EXP  1410
  870 FORMAT(//,  10X, 3HARC,  11X,  6HBRANCH,  8X,  10H TRAVERSAL,  3X,  EXP  1420
     1 15HEXPECTED NUMBER,  /,  5X,  4HTAIL,  5X,  4HHEAD,   7X,          EXP  1430
     2                6HPROB.  ,  10X,  5HPROB.  , 8X,                   EXP  1440
     3   9HOF ERRORS    )                                                EXP  1450
  880 FORMAT(   5X,   I3,   6X,   I3,    3X,              F10.4,         EXP  1460
     1        6X,  F10.4 ,  4X,  F10.2    )                             EXP  1470
  890 FORMAT( ////, 35H INITIAL EXPECTED NUMBER OF ERRORS=      ,         EXP  1480
     2  F10.2     )                                                      EXP  1490
  900 FORMAT( //,   47X,    11HCUMMULATIVE,    /,                       EXP  1500
     2  20X,  15HEXPECTED NUMBER,   10X,   15HEXPECTED NUMBER,/,         EXP  1510
     3  4X,  5HINPUT,    10X,   18HOF ERRORS DETECTED ,  7X,             EXP  1520
     4  18HOF ERRORS DETECTED, /,  (I7,  13X,  F10.2,  15X, F10.2 ))    EXP  1530
  910 FORMAT(  74H1GRAPH OF EXPECTED NUMBER OF ERRORS DETECTED -VS- NUMB  EXP  1540
     1ER OF INPUTS.                )                                     EXP  1550
  920 FORMAT(  48X,   12HINPUT NUMBER   )                                 EXP  1560
  930 FORMAT(   80H1GRAPH OF CUMMULATIVE EXPECTED NUMBER OF ERRORS DETEC  EXP  1570
     1TED -VS- NUMBER OF INPUTS.    )                                     EXP  1580
  940 FORMAT( 20H INPUTS SET EQUAL TO  ,  I5  )                           EXP  1590
  950 FORMAT( 38H EXPECTED NUMBER OF ERRORS IS NEGATIVE,     F10.5 )      EXP  1600
  960 FORMAT(1H1)                                                         EXP  1610
      END
```

```
C.....PLOP0010
C                                                                      PLOP0020
C     SUBROUTINE PLOTP/DPLTP                (CATEGORY J5)              PLOP0030
C                                                                      PLOP0040
C     PURPOSE                                                          PLOP0050
C        PRINTS GRAPHS ON THE STANDARD OUTPUT LINE PRINTER. AT THE USERS PLOP0060
C     OPTION, PLOTP/DPLTP USES ONE OF SEVERAL METHODS TO OBTAIN SCALE  PLOP0070
C     FACTORS.  THE SCALE FACTORS USED ARE PRINTED AT THE BOTTOM OF    PLOP0080
C     THE GRAPH.                                                       PLOP0090
C                                                                      PLOP0100
C                                                                      PLOP0110
C     CALLING SEQUENCE                                                 PLOP0120
C                                                                      PLOP0130
C        CALL PLOTP(X,Y,N,MODCUR)                                      PLOP0140
C                                                                      PLOP0150
C        CALL DPLTP(X,Y,N,MODCUR)                                      PLOP0160
C                                                                      PLOP0170
C     DESCRIPTION OF ARGUMENTS                                         PLOP0180
C                                                                      PLOP0190
C     X       VECTOR OF ABSCISSAE (REAL*4 WHEN CALLING PLOTP AND REAL*8 PLOP0200
C             WHEN CALLING DPLTP)                                      PLOP0210
C                                                                      PLOP0220
C     Y       VECTOR OF ASSOCIATED ORDINATES (REAL*4 WHEN CALLING PLOTP PLOP0230
C             AND REAL*8 WHEN CALLING DPLTP)                           PLOP0240
C                                                                      PLOP0250
C     N       NUMBER OF (X,Y) PAIRS                                    PLOP0260
C                                                                      PLOP0270
C     MODCUR CONTROLS THE NUMBER OF CURVES ON ONE GRAPH.  SEE BELOW    PLOP0280
C             FOR EXPLANATION OF MULTIPLE VALUES OF MODCUR.            PLOP0290
C                                                                      PLOP0300
C             =0, 4, OR  8  THERE IS ONLY ONE CURVE ON THIS GRAPH      PLOP0310
C             =1, 5, OR  9  THIS IS THE FIRST OF TWO OR MORE CURVES ON PLOP0320
C                           THIS GRAPH.                                PLOP0330
C             =2, 6, OR 10  THIS IS AN INTERMEDIATE CURVE ON THIS GRAPH. PLOP0340
C             =3, 7, OR 11  THIS IS THE LAST CURVE ON THIS GRAPH.      PLOP0350
C                                                                      PLOP0360
C     SCALING                                                          PLOP0370
```

```
C                                                                       PLOP0380
C          SCALING IS PERFORMED ONLY ON THE FIRST SET OF POINTS (WHEN   PLOP0390
C     MODCUR IS 0, 1, 4, 5, 8 OR 9.)  FOR A MULTIPLE CURVE GRAPH,       PLOP0400
C     THE USER SHOULD CALL THE SUBROUTINE INITIALLY WITH THE LARGEST    PLOP0410
C     CURVE TO INSURE OPTIMUM SCALING.                                  PLOP0420
C                                                                       PLOP0430
C          PLOTP/DPLTP USES ONE OF THREE OPTIONAL METHODS FOR           PLOP0440
C     DETERMINING THE APPROPRIATE SCALE FACTORS.                        PLOP0450
C                                                                       PLOP0460
C  METHOD 1:  PLOTP SCANS THE X AND Y ARRAYS TO DETERMINE THE MAXI-     PLOP0470
C             MUM AND MINIMUM VALUES.  SCALE FACTORS ARE CHOSEN SO      PLOP0480
C             THAT MAXIMUM X AND MIMIMUM X ARE SLIGHTLY WITHIN THE      PLOP0490
C             LEFT AND RIGHT, RESPECTIVELY OF THE GRAPH BOUNDARIES      PLOP0500
C             AND, SIMILARLY, THAT MAX Y AND MIN Y ARE SLIGHTLY         PLOP0510
C             BELOW AND ABOVE, RESPECTIVELY, THE GRAPH BOUNDARIES.      PLOP0520
C             UTPLOT IS THEN CALLED TO PLOT THE GRAPH.                  PLOP0530
C                                                                       PLOP0540
C  METHOD 1 IS USED WHEN MODCUR = 0, 1, 2, OR  3                        PLOP0550
C                                                                       PLOP0560
C  METHOD 2:  PLOTP DETERMINES THE MAXIMUM AND MINIMUM VALUES OF X      PLOP0570
C             AND Y AS IN METHOD 1. IF THE MINIMUM IS GREATER THAN      PLOP0580
C             ZERO, IT IS SET TO ZERO, AND IF THE MAXIMUM IS LESS       PLOP0590
C             THAN ZERO IT IS SET TO ZERO. THE MAXIMUM IS THEN          PLOP0600
C             ROUNDED UP TO THE NEXT HIGHEST VALUE CONTAINING 2         PLOP0610
C             SIGNIFICANT FIGUES AND THE MINIMUM IS ROUNDED DOWN TO     PLOP0620
C             THE NEXT LOWEST VALUE CONTAINING 2 SIGNIFICANT FIGURES.   PLOP0630
C             A RANGE IS THEN COMPUTED WHICH IS THE DIFFERENCE BE-      PLOP0640
C             TWEEN THE MAXIMUM AND THE MINIMUM. THE RANGE IS AD-       PLOP0650
C             JUSTED UNTIL, IN THE X-DIRECTION, IT IS A MULTIPLE OF     PLOP0660
C             4, AND IN THE Y-DIRECTION, IT IS A MULTIPLE OF 6.         PLOP0670
C             ROUNDED MAXIMUM AND MINIMUMS ARE THEN USED TO CALL        PLOP0680
C             SUBROUTINE UTPLOT WHICH PLOTS THE GRAPH.                  PLOP0690
C             THIS ALGORITHM IS USED TO "RATIONALIZE" THE AXES AND      PLOP0700
C             SCALE FACTORS.                                            PLOP0710
C                                                                       PLOP0720
C  METHOD 2 IS USED WHEN MODCUR = 4, 5, 6, OR 7                         PLOP0730
```

```
C                                                                       PLOP0740
C     METHOD 3:  THIS IS THE SAME AS METHOD 2 EXCEPT WHEN ALL VALUES    PLOP0750
C                OF X OR Y ARE POSITIVE OR NEGATIVE (NO ZERO), THE      PLOP0760
C                ORIGIN OF X OR Y (RESPECTIVELY) DOES NOT APPEAR ON THE PLOP0770
C                GRAPH.                                                 PLOP0780
C                                                                       PLOP0790
C     METHOD 3 IS USED WHEN MODCUR = 8, 9, 10, OR 11                    PLOP0800
C                                                                       PLOP0810
C                                                                       PLOP0820
C          IN EITHER METHOD, EACH PRINT POSITION IN THE X-DIRECTION     PLOP0830
C     WILL BE EQUAL TO (XMAX-XMIN)/80. AND EACH PRINT POSITION IN THE   PLOP0840
C     Y-DIRECTION WILL BE EQUAL TO (YMAX-YMIN)/60.                      PLOP0850
C                                                                       PLOP0860
C     GRID LABELLING                                                    PLOP0870
C                                                                       PLOP0880
C          THE DATA TO BE GRAPHED WILL BE FIT INTO AN 80 COLUMN BY      PLOP0890
C     BY 60 ROW GRID. THE GRID WILL BE LABELLED AS FOLLOWS:             PLOP0900
C                                                                       PLOP0910
C     IN THE X DIRECTION (COLUMN-WISE), THERE WILL BE 5 VALUES:         PLOP0920
C     THE MAXIMUM, THE MINIMUM, AND 3 INTERMEDIATE EQUALLY SPACED       PLOP0930
C     VALUES.                                                           PLOP0940
C                                                                       PLOP0950
C     IN THE Y DIRECTION (ROW-WISE) THERE WILL BE 7 VALUES: THE MAXI-   PLOP0960
C     MUM, THE MINIMUM, AND 5 INTERMEDIATE EQUALLY SPACED VALUES.       PLOP0970
C                                                                       PLOP0980
C     IF THE LABELS HAVE A VALUE BETWEEN 1. AND 10**8, THEY WILL BE     PLOP0990
C     PRINTED IN AN F11.2 FORMAT, OTHERWISE THEY WILL BE PRINTED IN A   PLOP1000
C     1PE10.3 FORMAT.                                                   PLOP1010
C                                                                       PLOP1020
C     PLOTTING                                                          PLOP1030
C                                                                       PLOP1040
C          FOUR CHARACTERS ARE USED FOR PLOTTING CURVES, ".","+","*",   PLOP1050
C     AND "X". WHEN MORE THAN ONE CURVE APPEARS ON A GRAPH, EACH IS     PLOP1060
C     USED IN TURN IN THE ABOVE ORDER.  IF MORE THAN 4 CURVES ARE       PLOP1070
C     PLOTTED THE ABOVE CYCLE OF CHARACTERS IS REPEATED.  IF A NEW      PLOP1080
C     CURVE IS TO BE PLACED IN THE PLOTTING GRID WHERE AN OLD CURVE     PLOP1090
C     EXISTS, THE NEW CURVE CHARACTER REPLACES THE OLD ONE.  THUS, IF   PLOP1100
C     3 IDENTICAL CURVES ARE PLOTTED, THEY WILL APPEAR AS ONE CURVE     PLOP1110
C     COMPOSED OF "*"'S.                                                PLOP1120
```

```
C                                                                       PLOP1130
C     MESSAGES                                                          PLOP1140
C                                                                       PLOP1150
C         UNDER CERTAIN CIRCUMSTANCES, A PLOT WILL NOT BE OUTPUT AND    PLOP1160
C     ONE OF THE FOLLOWING SELF-EXPLANATORY MESSAGES WILL BE            PLOP1170
C     PRINTED ON THE STANDARD OUTPUT IN PLACE OF THE PLOT.              PLOP1180
C                                                                       PLOP1190
C     THE FOLLOWING MESSAGES ARE PRODUCED BY PLOTP/DPLTP:               PLOP1200
C                                                                       PLOP1210
C     "ALL X VALUES 0. OR THE SAME VALUE. CANNOT SETUP PLOT GRID        PLOP1220
C      CHECK MAX AND MIN X FOR INITIAL CALL TO PLOTP/DPLTP."            PLOP1230
C                                                                       PLOP1240
C     "ALL Y VALUES 0. OR THE SAME VALUE. CANNOT SETUP PLOT GRID        PLOP1250
C      CHECK MAX AND MIN Y FOR INITIAL CALL TO PLOTP/DPLTP."            PLOP1260
C                                                                       PLOP1270
C     "GRID NOT SETUP WHEN PLOT INITIALIZED. NO PLOT UNTIL GRID         PLOP1280
C      PROPERLY SETUP."                                                 PLOP1290
C                                                                       PLOP1300
C     THE FOLLOWING MESSAGES ARE PRODUCED BY UTPLOT:                    PLOP1310
C                                                                       PLOP1320
C     "ALL X VALUES = 0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN X      PLOP1330
C      WHEN MODCUR = 0 OR 1."                                           PLOP1340
C                                                                       PLOP1350
C     "ALL Y VALUES = 0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN Y      PLOP1360
C      WHEN MODCUR = 0 OR 1."                                           PLOP1370
C                                                                       PLOP1380
C     "GRID NOT SETUP WHEN MODCUR LAST 0 OR 1. NO PLOT UNTIL            PLOP1390
C      PROPERLY SETUP."                                                 PLOP1400
C                                                                       PLOP1410
C     NOTE                                                              PLOP1420
C                                                                       PLOP1430
C         THE USER IS EXPECTED TO PROVIDE THE NECESSARY CARRIAGE        PLOP1440
C     CONTROLS TO PLACE THE GRAPH PROPERLY ON THE PAGE.  BEFORE         PLOP1450
C     CALLING PLOTP THE USER SHOULD ISSUE A PRINT STATEMENT WHICH       PLOP1460
```

```
C        EJECTS A PAGE SO THAT THE GRAPH WILL BE PLOTTED AT THE TOP      PLOP1470
C        OF THE NEXT PAGE. A TITLE CAN BE PRINTED AT THE BOTTOM OF       PLOP1480
C        THE GRAPH BY ISSUING A PRINT STATEMENT RIGHT AFTER CALLING      PLOP1490
C        THE SUBROUTINE.                                                 PLOP1500
C                                                                        PLOP1510
C     SUBROUTINES REQUIRED                                               PLOP1520
C           PLOTP/DPLTP CALLS SUBROUTINE UTPLOT WHICH HAS BEEN COMPILEDPLOP1530
C        AND ADDED TO MPSLIB UNDER OS.                                   PLOP1540
C                                                                        PLOP1550
C     PROGRAMMER                                                         PLOP1560
C                                                                        PLOP1570
C        S. KLINE/ G. BARKSDALE (1968)                                   PLOP1580
C        MODIFIED BY S.D. RANEY 12/71,8/74                               PLOP1590
C                                                                        PLOP1600
C.......................................................................PLOP1610
C                                                                        PLOP1620
      SUBROUTINE PLOTP(X,Y,NN,MODCUR  )                                  PLOP1630
      DIMENSION X( 1), Y( 1), RANGE(4)                                   PLOP1640
      EQUIVALENCE (RANGE(1),XMAX), (RANGE(2),XMIN),(RANGE(3),YMAX),      PLOP1650
     1           (RANGE(4),YMIN)                                         PLOP1660
      DATA JERR/0/                                                       PLOP1670
C                                                                        PLOP1680
      KKZ=1                                                              PLOP1690
      GO TO 20                                                           PLOP1700
      ENTRY DPLTP(X,Y,NN,MODCUR)                                         PLOP1710
      KKZ=2                                                              PLOP1720
   20 KN=IABS(NN)                                                        PLOP1730
      KDATA=KN*KKZ                                                       PLOP1740
      MMC=MOD(MODCUR,4)                                                  PLOP1750
      IF(MMC.GT.1.AND.JERR.GT.0) GO TO 885                               PLOP1760
      ISCT=MODCUR/4+1                                                    PLOP1770
      IF(NN.LT.0) ISCT=2                                                 PLOP1780
      IF(MMC   .GT.1) GO TO 5                                            PLOP1790
C                                                                        PLOP1800
C     FIND MAX & MIN FOR SCALE COMPUTATIONS                              PLOP1810
C                                                                        PLOP1820
      JERR=0                                                             PLOP1830
      XMAX=-1.E20                                                        PLOP1840
      XMIN=1.E20                                                         PLOP1850
      YMAX=-1.E20                                                        PLOP1860
      YMIN=1.E20                                                         PLOP1870
```

```
      DO 1 I=1,KDATA,KKZ                                                PLOP1880
      IF(X(I).LT.XMAX) GO TO 6                                          PLOP1890
      XMAX=X(I)                                                         PLOP1900
    6 IF(X(I).GT.XMIN) GO TO 7                                          PLOP1910
      XMIN=X(I)                                                         PLOP1920
    7 IF(Y(I).LT.YMAX) GO TO 8                                          PLOP1930
      YMAX=Y(I)                                                         PLOP1940
    8 IF(Y(I).GT.YMIN) GO TO 1                                          PLOP1950
      YMIN=Y(I)                                                         PLOP1960
    1 CONTINUE                                                          PLOP1970
C                                                                       PLOP1980
CC    IF NOT AUTOSCALE GO TO CALL UTPLOT                                PLOP1990
C                                                                       PLOP2000
      IF(XMAX.EQ.XMIN) GO TO 887                                        PLOP2010
      IF(YMAX.EQ.YMIN) GO TO 889                                        PLOP2020
      IF(ISCT.EQ.1) GO TO 5                                             PLOP2030
C                                                                       PLOP2040
CC    COMPUTE X-SCALE & NEW XMAX AND XMIN                               PLOP2050
C                                                                       PLOP2060
      CALL PSCALE(XMAX,XMIN,4,ISCT)                                     PLOP2070
C                                                                       PLOP2080
CC    COMPUTE Y-SCALE & NEW YMAX AND YMIN                               PLOP2090
C                                                                       PLOP2100
      CALL PSCALE(YMAX,YMIN,6,ISCT)                                     PLOP2110
C                                                                       PLOP2120
CC    PLOT CURVE                                                        PLOP2130
C                                                                       PLOP2140
    5 CALL UTPLOT(X,Y,KN,RANGE,KKZ,MMC)                                 PLOP2150
      IF(MMC    . EQ.1.OR.MMC    .EQ.2) RETURN                         PLOP2160
C                                                                       PLOP2170
CC    PRINT SCALES WHEN LAST CURVE PLOTTED                              PLOP2180
C                                                                       PLOP2190

      XS=(XMAX-XMIN)/80.                                                PLOP2200
      YS=(YMAX-YMIN)/60.                                                PLOP2210
      WRITE(6,100) XS,YS                                                PLOP2220
  100 FORMAT('  15X,'X-SCALE:   "*"=',E10.3,' UNITS'//                  PLOP2230
     1       15X,'Y-SCALE:   "*"=',E10.3,' UNITS')                      PLOP2240
      RETURN                                                            PLOP2250
  889 WRITE(6,888)                                                      PLOP2260
  888 FORMAT(' ALL Y VALUES=0. OR THE SAME VALUE.  CANNOT SETUP PLOT GRI PLOP2270
     1D. CHECK MAX AND MIN Y FOR INITIAL CALL TO PLOTP/DPLTP')          PLOP2280
      JERR=10                                                           PLOP2290
      RETURN                                                            PLOP2300
  887 WRITE(6,886)                                                      PLOP2310
  886 FORMAT(' ALL X VALUES=0. OR THE SAME VALUE.  CANNOT SETUP PLOT GRI PLOP2320
     1D. CHECK MAX AND MIN X FOR INITIAL CALL TO PLOTP/DPLTP')          PLOP2330
      JERR=10                                                           PLOP2340
      RETURN                                                            PLOP2350
  885 WRITE(6,884)                                                      PLOP2360
  884 FORMAT(' GRID NOT SETUP WHEN PLOT INITIALIZED.    NO PLOT UNTIL GR PLOP2370
     1ID PROPERLY SETUP')                                               PLOP2380
      RETURN                                                            PLOP2390
      END                                                               PLOP2400
```

```
      SUBROUTINE PSCALE(XMAX,XMIN,IDIV,ISCT)                            PLOP2410
      DIV=IDIV                                                          PLOP2420
C     ROUND MAXIMUM TO NEXT HIGHEST 2 SIG FIGS                          PLOP2430
      IF(ISCT.LT.3) XMAX=AMAX1(0.,XMAX)                                 PLOP2440
      CALL ROUND (XMAX,IMX,FMX)                                         PLOP2450
      IF(ABS(XMX).LE.1.E-7) XMX=0.0                                     PLOP2460
      IMX=IMX-1                                                         PLOP2470
      FMX=FMX*10.                                                       PLOP2480
    3 XMX=FMX*10.**IMX                                                  PLOP2490
      IF(XMX.GE.XMAX) GO TO 2                                           PLOP2500
      FMX=FMX+1.                                                        PLOP2510
      IMM=FMX                                                           PLOP2520
      FMX=IMM                                                           PLOP2530
      GO TO 3                                                           PLOP2540
C     ROUND MINIMUM TO NEXT LOWEST 2 SIG FIGS                           PLOP2550
    2 IF(ISCT.LT.3) XMIN=AMIN1(0.,XMIN)                                 PLOP2560
      CALL ROUND (XMIN,IMN,FMN)                                         PLOP2570
      IMN=IMN-1                                                         PLOP2580
      FMN=FMN*10.                                                       PLOP2590
   14 XMN=FMN*10.**IMN                                                  PLOP2600
      IF(XMIN.GE.XMN) GO TO 11                                          PLOP2610
      FMN=FMN-1.                                                        PLOP2620
      IMM=FMN                                                           PLOP2630
      FMN=IMM                                                           PLOP2640
      GO TO 14                                                          PLOP2650
C     ROUND MAX & MIN TO 1. OR .1 IF RANGE LARGE                        PLOP2660
   11 IF(XMN.LE.1.E-7) XMN=0.                                           PLOP2670
      XSC=XMX-XMN                                                       PLOP2680
      IM=0                                                              PLOP2690
      SM=1.                                                             PLOP2700
    9 IF(XSC/DIV.LE.SM) GO TO 12                                        PLOP2710
      IF(ABS(XMN).LT.SM.AND.ABS(XMN).GT.0.) XMN=SIGN(SM,XMN)            PLOP2720
      IF(ABS(XMX).LT.SM.AND.ABS(XMX).GT.0.) XMX=SIGN(SM,XMX)            PLOP2730
   12 IF(IM.GT.0) GO TO 19                                              PLOP2740
      SM=.1                                                             PLOP2750
      IM=IM+1                                                           PLOP2760
      GO TO 9                                                           PLOP2770
```

```
C       ROUND RANGE (MAX-MIN) TO 2 SIG FIGS                                  PLOP2780
   19   XSC=XMX-XMN                                                          PLOP2790
        CALL RCUND (XSC,IS10,FACTX)                                          PLOP2800
C       FIND FACTOR WHICH IS MULTIPLE OF IDIV                                PLOP2810
        FACTX=FACTX*10.                                                      PLOP2820
        OFAC=FACTX                                                           PLOP2830
        IS10=IS10-1                                                          PLOP2840
        IFX=FACTX                                                            PLOP2850
        FACTX=IFX                                                            PLOP2860
   20   IF(MOD(IFX,IDIV).EQ.0.AND.FACTX.GE.OFAC) GO TO 10                    PLOP2870
        IFX=IFX+1                                                            PLOP2880
        FACTX=IFX                                                            PLOP2890
        GO TO 20                                                             PLOP2900
   10   IF(IDIV.GT.4) GO TO 15                                               PLOP2910
C       IF X SCALE BETWEEN 8. AND 10., ROUND TO 10.                          PLOP2920
        FFX=ABS(FACTX/10.)                                                   PLOP2930
        IF(FFX.GT.8..AND.FFX.LT.10.) FFX=10.                                 PLOP2940
        IF(FACT X.LT.0.) FXX=-10.                                            PLOP2950
        FACTX=FFX*10.                                                        PLOP2960
   15   XSC=FACTX*10.**IS10                                                  PLOP2970
C       COMPUTE NEW MAX & MIN FROM ROUNDED SCALE                             PLOP2980
        IF(XM N*XM X.NE.0.) GO TO 4                                          PLOP2990
        IF(XM N.LT.0.) XMIN=-XSC                                             PLOP3000
        IF(XM X.GT.0.) XMAX=XSC                                              PLOP3010
        RETURN                                                               PLOP3020
    4   XMAX=XSC+XMN                                                         PLOP3030
        XMIN=XMN                                                             PLOP3040
        RETURN                                                               PLOP3050
        END                                                                  PLOP3060
```

```
      SUBROUTINE ROUND(ANUM,IS,FACT)                                    PLOP3070
C                                                                       PLOP3080
C     EXPRESS ANUM IN SCIENTIFIC NOTATION WHERE                         PLOP3090
C           ANUM=FACT*10.**IS WHERE FACT IS BETWEEN 1. AND 9.9          PLOP3100
C                                                                       PLOP3110
      IF(ANUM.EQ.0.) GO TO 15                                           PLOP3120
      BNUM=ANUM                                                         PLOP3130
      IF(BNUM.LT.0.) BNUM=-BNUM                                         PLOP3140
      IS=ALOG(BNUM)*.43429448                                           PLOP3150
      FACT=BNUM/10.**IS                                                 PLOP3160
C     FIND POWER OF 10                                                  PLOP3170
      IDD=-3                                                            PLOP3180
      R2=0                                                              PLOP3190
      DO 10 II=1,5                                                      PLOP3200
      IDD=IDD+1                                                         PLOP3210
      R1=R2                                                             PLOP3220
      R2=10.**(IDD+1)                                                   PLOP3230
      IF(FACT.GE.R1.AND.FACT.LT.R2) GO TO 8                             PLOP3240
   10 CONTINUE                                                          PLOP3250
    8 FACT=FACT*10.**(-IDD)                                             PLOP3260
      IS=IS+IDD                                                         PLOP3270
C     ROUND MANTISSA TO 2 SIG FIGS                                      PLOP3280
      IFAC=FACT*10.+.05                                                 PLOP3290
      FACT=IFAC                                                         PLOP3300
      FACT=FACT/10.                                                     PLOP3310
      IF(FACT.LT.10.) GO TO 20                                          PLOP3320
C     SET TO 1 IF LESS THAN 10.                                         PLOP3330
      FACT=1.                                                           PLOP3340
      IS=IS+1                                                           PLOP3350
C     IF INPUT NEGATIVE, SET MANTISSA NEGATIVE                          PLOP3360
   20 IF(ANUM.LT.0.) FACT=-FACT                                         PLOP3370
      RETURN                                                            PLOP3380
C     SET TO 0. IF 0.                                                   PLOP3390
   15 FACT=0.                                                           PLOP3400
      IS=0                                                              PLOP3410
      RETURN                                                            PLOP3420
      END                                                               PLOP3430
```

```
C.....UTPL0010
C                                                                      UTPL0020
C      SUBROUTINE UTPLOT                                               UTPL0030
C                                                                      UTPL0040
C      PURPOSE                                                         UTPL0050
C                                                                      UTPL0060
C          PRINTS GRAPHS ON THE STANDARD OUTPUT PRINTER                UTPL0070
C                                                                      UTPL0080
C      FEATURES                                                        UTPL0090
C                                                                      UTPL0100
C      1) FULL CONTROL OVER SCALING                                    UTPL0110
C      2) ABILITY TO PLOT SINGLE OR DOUBLE PRECISION VECTORS           UTPL0120
C                                                                      UTPL0130
C      CALLING SEQUENCE                                                UTPL0140
C                                                                      UTPL0150
C      CALL UTPLOT(X,Y,N,RANGE,K,MODCUR)                               UTPL0160
C                                                                      UTPL0170
C      DESCRIPTION OF ARGUMENTS                                        UTPL0180
C                                                                      UTPL0190
C      X       VECTOR OF ABSCISSAE                                     UTPL0200
C                                                                      UTPL0210
C      Y       VECTOR OF ASSOCIATED ORDINATES                          UTPL0220
C                                                                      UTPL0230
C      N       NUMBER OF (X,Y) PAIRS                                   UTPL0240
C                                                                      UTPL0250
C      RANGE   4 WORD SCALING VECTOR WHERE                             UTPL0260
C              RANGE(1)= MAXIMUM X TO BE PLOTTED                       UTPL0270
C              RANGE(2)= MINIMUM X TO BE PLOTTED                       UTPL0280
C              RANGE(3)= MAXIMUM Y TO BE PLOTTED                       UTPL0290
C              RANGE(4)= MINIMUM Y TO BE PLOTTED                       UTPL0300
C                                                                      UTPL0310
C              ALL (X,Y) POINTS OUTSIDE THE ABOVE RANGE WILL BE PLOTTEDUTPL0320
C              IN THE BORDER OF THE GRAPH.                             UTPL0330
C                                                                      UTPL0340
C      K       EVERY KTH ELEMENT OF X & Y WILL BE PLOTTED, E.G.,       UTPL0350
C              FOR REAL*4 DATA (SINGLE PRECISION) K=1                  UTPL0360
C              FOR REAL*8 DATA (DOUBLE PRECISION) K=2.                 UTPL0370
C                                                                      UTPL0380
C      MODCUR CONTROLS THE NUMBER OF CURVES ON ONE GRAPH               UTPL0390
C             =0 THERE IS ONLY 1 CURVE ON THIS GRAPH                   UTPL0400
C             =1 THIS IS THE FIRST OF TWO OR MORE CURVES ON THIS GRAPH  UTPL0410
C             =2 THIS IS AN INTERMEDIATE CURVE ON THIS GRAPH           UTPL0420
C             =3 THIS IS THE LAST CURVE ON THIS GRAPH                  UTPL0430
C                                                                      UTPL0440
```

```
C
C     SCALING                                                          UTPL0450
C                                                                      UTPL0460
C         SCALING IS PERFORMED ONLY ON THE FIRST SET OF POINTS (WHEN   UTPL0470
C     MODCUR = 0 OR 1.) ARRAY RANGE IS USED TO SET UP THE SCALE FAC-   UTPL0480
C     TORS AND NEED ONLY BE DEFINED FOR THE FIRST CALL TO UTPLOT.      UTPL0490
C                                                                      UTPL0500
C     GRID LABELLING                                                   UTPL0510
C                                                                      UTPL0520
C         THE DATA TO BE GRAPHED WILL BE FIT INTO AN 80 COLUMN BY      UTPL0530
C     BY 60 ROW GRID. THE GRID WILL BE LABELLED THUSLY:                UTPL0540
C                                                                      UTPL0550
C     IN THE X DIRECTION (COLUMN-WISE), THERE WILL BE 5 VALUES:        UTPL0560
C     THE MAXIMUM, THE MINIMUM, AND 3 INTERMEDIATE AT INCREMENTS OF    UTPL0570
C                                                                      UTPL0590
C     (RANGE(2)-RANGE(1))/4. FROM THE MINIMUM.                         UTPL0580
C     IN THE Y DIRECTION (ROW-WISE) THERE WILL BE 7 VALUES: THE MAXI-  UTPL0600
C     MUM, THE MINIMUM, AND 5 INTERMEDIATE AT INCREMENTS OF            UTPL0610
C     (RANGE(4)-RANGE(3))/6. FROM THE MINIMUM.                         UTPL0620
C                                                                      UTPL0630
C     IF THE LABELS HAVE A VALUE BETWEEN 1. AND 10**8, THEY WILL BE    UTPL0640
C     PRINTED IN AN F11.2 FORMAT, OTHERWISE THEY WILL BE PRINTED IN A  UTPL0650
C     1PE10.3 FORMAT.                                                  UTPL0660
C                                                                      UTPL0670
C     PLOTTING                                                         UTPL0680
C                                                                      UTPL0690
C         FOUR CHARACTERS ARE USED FOR PLOTTING CURVES, ".","+","*",   UTPL0700
C     AND "X". WHEN MORE THAN 4 CURVES ARE PLOTTED THE CHARACTERS      UTPL0710
C     ARE USED REPEATEDLY.  IF A NEW CURVE IS TO BE PLACED IN THE      UTPL0720
C     PLOTTING GRID WHERE AN OLD CURVE EXISTS, THE NEW CURVE REPLACES  UTPL0730
```

```
C        THE OLD ONE. THUS, IF 3 IDENTICAL CURVES ARE PLOTTED, THEY WILL  UTPL0740
C        APPEAR AS ONE CURVE COMPOSED OF "*"'S.                           UTPL0750
C                                                                         UTPL0760
C     MESSAGES                                                            UTPL0770
C                                                                         UTPL0780
C           UNDER CERTAIN CIRCUMSTANCES, A PLOT WILL NOT BE OUTPUT AND    UTPL0790
C        ONE OF THE FOLLOWING MESSAGES WILL BE PRINTED ON THE STANDARD    UTPL0800
C        OUTPUT IN PLACE OF THE PLOT.                                     UTPL0810
C                                                                         UTPL0820
C        "ALL Y-VALUES=0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN Y       UTPL0830
C         WHEN MODCUR=0 OR 1."                                            UTPL0840
C                                                                         UTPL0850
C        "ALL X VALUES=0. CANNOT SETUP PLOT GRID. CHECK MAX AND MIN X     UTPL0860
C         WHEN MODCUR=0 OR 1."                                            UTPL0870
C                                                                         UTPL0880
C        "GRID NOT SETUP WHEN MODCUR LAST 0 OR 1. NO PLOT UNTIL GRID      UTPL0890
C         PROPERLY SETUP."                                                UTPL0900
C                                                                         UTPL0910
C     NOTE                                                                UTPL0920
C                                                                         UTPL0930
C           THE USER IS EXPECTED TO PROVIDE THE NECESSARY CARRIAGE        UTPL0940
C        CONTROLS TO PLACE THE GRAPH PROPERLY ON THE PAGE.  BEFORE        UTPL0950
C        CALLING UTPLOT THE USER SHOULD ISSUE A PRINT STATEMENT WHICH     UTPL0960
C        EJECTS A PAGE SO THAT THE GRAPH WILL BE PLOTTED AT THE TOP       UTPL0970
C        OF THE NEXT PAGE. A TITLE CAN BE PRINTED AT THE BOTTOM OF        UTPL0980
C        THE GRAPH BY ISSUING A PRINT STATEMENT RIGHT AFTER CALLING       UTPL0990
C        THE SUBROUTINE.                                                  UTPL1000
C                                                                         UTPL1010
C.........................................................................UTPL1020
C                                                                         UTPL1030
      SUBROUTINE UTPLOT (X   ,Y   ,NDATA,RANGE,KKZ,MODCUR)               UTPL1040
      DIMENSION GRID(61,81),XSCALE(5),YSCALE(7)                           UTPL1050
      DIMENSION X   (1),Y   (1),RANGE(4)                                  UTPL1060
      INTEGER*2 GRID,BLANK,DOT,XCHAR(4)/1H.,1H+,1H*,1HX/                  UTPL1070
      DATA DOT,BLANK/Z4B40,Z4040/                                         UTPL1080
      KDATA=NDATA*KKZ                                                     UTPL1090
      IF(MODCUR.GT.1) GO TO 444                                           UTPL1100
C                                                                         UTPL1110
C     GRID IS THE MATRIX USED TO PLOT THE POINTS                          UTPL1120
```

```
C                                                                       UTPL1130
      IERR=0                                                            UTPL1140
      XMAX=RANGE(1)                                                     UTPL1150
      XMIN=RANGE(2)                                                     UTPL1160
      YMAX=RANGE(3)                                                     UTPL1170
      YMIN=RANGE(4)                                                     UTPL1180
C                                                                       UTPL1190
C     CHECKING X AND Y POINTS AND PLOTTING THOSE OUT OF RANGE           UTPL1200
C     AT THE MARGIN                                                     UTPL1210
C                                                                       UTPL1220
      DO 30 I=1,KDATA,KKZ                                               UTPL1230
      IF(X(I).GT.XMAX.OR.X(I).LT.XMIN.OR.Y(I).GT.YMAX.OR.Y(I).LT.YMIN)  UTPL1240
     1  IERR=IERR+1                                                     UTPL1250
      IF(X(I).LE.XMAX) GO TO 205                                        UTPL1260
      X  (I)=XMAX                                                       UTPL1270
      GOTO 210                                                          UTPL1280
  205 IF(X(I).GE.XMIN) GO TO 210                                        UTPL1290
      X  (I)=XMIN                                                       UTPL1300
  210 IF(Y(I).LE.YMAX) GO TO 215                                        UTPL1310
      Y  (I)=YMAX                                                       UTPL1320
      GOTO 30                                                           UTPL1330
  215 IF(Y(I).GE.YMIN) GO TO 30                                         UTPL1340
      Y  (I)=YMIN                                                       UTPL1350
C                                                                       UTPL1360
   30 CONTINUE                                                          UTPL1370
C                                                                       UTPL1380
C     PLOTTING X AND Y AXIS , IF NECESSARY                              UTPL1390
C                                                                       UTPL1400
      JERR=0                                                            UTPL1410
      XRANGE=XMAX-XMIN                                                  UTPL1420
      YRANGE=YMAX-YMIN                                                  UTPL1430
      IF (YRANGE.NE.0.)  GO TO 298                                      UTPL1440
      IF(YMIN.EQ.0.) GO TO 889                                          UTPL1450
      YMIN=0.                                                           UTPL1460
```

```
      YRANGE=YMAX                                                       UTPL1470
      GO TO 299                                                         UTPL1480
  298 IF (XRANGE.NE.0.)  GO TO 299                                      UTPL1490
      IF(XMIN.EQ.0.) GO TO 887                                          UTPL1500
      XMIN=0.                                                           UTPL1510
      XRANGE=XMAX                                                       UTPL1520
C                                                                       UTPL1530
C     BLANKING OUT MATRIX-(GRID)                                        UTPL1540
C                                                                       UTPL1550
  299 DO 300 I=1,61                                                     UTPL1560
      DO 301 JJ=1,81                                                    UTPL1570
301   GRID(I,JJ)=BLANK                                                  UTPL1580
  300 CONTINUE                                                          UTPL1590
      IF(XMAX*XMIN.GE.0.) GO TO 222                                     UTPL1600
      IYAXIS=80.*(-XMIN)/XRANGE+1.5                                     UTPL1610
      DO 40 I=1,61                                                      UTPL1620
   40 GRID(I,IYAXIS)=DOT                                                UTPL1630
  222 IF(YMAX*YMIN.GE.0.) GO TO 333                                     UTPL1640
      IXAXIS=60.*YMAX/YRANGE+1.5                                        UTPL1650
      DO 60 I=1,81                                                      UTPL1660
   60 GRID(IXAXIS,I)=DOT                                                UTPL1670
C                                                                       UTPL1680
C     COMPUTE PROPER SCALE NUMBERS                                      UTPL1690
C                                                                       UTPL1700
  333 XINCR=XRANGE/4.                                                   UTPL1710
      YINCR=YRANGE/6.                                                   UTPL1720
      XSCALE(1)=XMAX                                                    UTPL1730
      XSCALE(5)=XMIN                                                    UTPL1740
      DO 80 I=2,4                                                       UTPL1750
      XSCALE(I)=XSCALE(I-1)-XINCR                                       UTPL1760
      IF(ABS(XSCALE(I)).LT.1.E-4) XSCALE(I)=0.                          UTPL1770
   80 CONTINUE                                                          UTPL1780
      YSCALE(1)=YMAX                                                    UTPL1790
      YSCALE(7)=YMIN                                                    UTPL1800
      DO 81 I=2,6                                                       UTPL1810
      YSCALE(I)=YSCALE(I-1)-YINCR                                       UTPL1820
      IF(ABS(YSCALE(I)).LT.1.E-4) YSCALE(I)=0.                          UTPL1830
   81 CONTINUE                                                          UTPL1840
      DO 85 II=1,2                                                      UTPL1850
      JJ=6-II                                                           UTPL1860
      XT=XSCALE(JJ)                                                     UTPL1870
      XSCALE(JJ)=XSCALE(II)                                             UTPL1880
   85 XSCALE(II)=XT                                                     UTPL1890
```

```
C                                                                       UTPL1900
C                                                                       UTPL1910
C     PLACING POINTS IN THEIR PROPER GRID POSITIONS                     UTPL1920
C                                                                       UTPL1930
  444 IF(MODCUR.LT.2) JSET=0                                            UTPL1940
      IF(JERR.GT.0) GO TO 885                                           UTPL1950
      JSET=JSET+1                                                       UTPL1960
      IF(JSET.GT.4) JSET=1                                              UTPL1970
      DO 700 I=1,KDATA,KKZ                                              UTPL1980
      IPTX=60.*(YMAX-Y(I))/YRANGE+1.5                                   UTPL1990
      IPTY=80.*(X(I)-XMIN)/XRANGE+1.5                                   UTPL2000
      IF(IPTX.GT.61.OR.IPTY.GT.81) GO TO 70                             UTPL2010
      IF(IPTX.LE.0.OR.IPTY.LE.0)GO TO 70                                UTPL2020
      GRID(IPTX,IPTY) = XCHAR(JSET)                                     UTPL2030
      GO TO 700                                                         UTPL2040
   70 IERR=IERR+1                                                       UTPL2050
  700 CONTINUE                                                          UTPL2060
C     OUTPUT SECTION WITH GRAPH                                         UTPL2070
C                                                                       UTPL2080
      IF(MODCUR.EQ.1.OR.MODCUR.EQ.2) RETURN                             UTPL2090
      AXR=ABS(XRANGE)                                                   UTPL2100
      AYR=ABS(YRANGE)                                                   UTPL2110
      IF(AXR.LT.1.E+8.AND.AXR.GE..95) GO TO 400                         UTPL2120
      WRITE(6,17) XSCALE                                                UTPL2130
   17 FORMAT(12X,1PE10.3,4(10X,E10.3)/15X,'**',8('+**********'),'+**')  UTPL2140
      GO TO 401                                                         UTPL2150
  400 WRITE(6,117) XSCALE                                               UTPL2160
  117 FORMAT( 8X,F11.2,4( 9X,F11.2)/15X,'**',8('+**********'),'+**')    UTPL2170
  401 II=1                                                              UTPL2180
      DO 101 IK=1,61                                                    UTPL2190

      IF(MOD(IK-1,10).NE.0) GO TO 92                                    UTPL2200
      IF(AYR.LT.1.E+8.AND.AYR.GE..95) GO TO 404                         UTPL2210
      WRITE(6,18) YSCALE(II),(GRID(IK,IX),IX=1,81),YSCALE(II)           UTPL2220
   18 FORMAT(3X,1PE10.3,2X,1H+,1X,81A1,1X,1H+,2X,E10.3)                 UTPL2230
      GO TO 405                                                         UTPL2240
  404 WRITE(6,118) YSCALE(II),(GRID(IK,IX),IX=1,81),YSCALE(II)          UTPL2250
  118 FORMAT(2X,F11.2,'  + ',81A1,' +  ',F11.2)                         UTPL2260
  405 II=II+1                                                           UTPL2270
      GO TO 101                                                         UTPL2280
   92 WRITE(6,19) (GRID(IK,IX),IX=1,81)                                 UTPL2290
   19 FORMAT(15X,'* ',81A1,' *')                                        UTPL2300
  101 CONTINUE                                                          UTPL2310
      IF(AXR.LT.1.E+8.AND.AXR.GE..95) GO TO 402                         UTPL2320
      WRITE(6,22) XSCALE                                                UTPL2330
   22 FORMAT(15X,'**',8('+**********'),'+**'/12X,1PE10.3,4(10X,E10.3),//)UTPL2340
      GO TO 403                                                         UTPL2350
  402 WRITE(6,217) XSCALE                                               UTPL2360
  217 FORMAT(15X,'**',8('+**********'),'+**'/ 8X,F11.2,4( 9X,F11.2),//)  UTPL2370
  403 IF(IERR.GT.0) WRITE(6,20) IERR                                    UTPL2380
   20 FORMAT(10X 'NUMBER OF POINTS OUT OF RANGE =', I4)                 UTPL2390
 1000 RETURN                                                            UTPL2400
C                                                                       UTPL2410
  889 WRITE(6,888)                                                      UTPL2420
  888 FORMAT(' ALL Y VALUES=0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN YUTPL2430
     1 WHEN MODCUR=0 OR 1.')                                            UTPL2440
      JERR=10                                                           UTPL2450
      RETURN                                                            UTPL2460
  887 WRITE(6,886)                                                      UTPL2470
  886 FORMAT(' ALL X VALUES=0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN XUTPL2480
     1 WHEN MODCUR=0 OR 1.')                                            UTPL2490
      JERR=10                                                           UTPL2500
      RETURN                                                            UTPL2510
  885 WRITE(6,884)                                                      UTPL2520
  884 FORMAT(' GRID NOT SETUP WHEN MODCUR LAST 0 OR 1. NO PLOT UNTIL GRIUTPL2530
```

```
 ERROR DETECTION MODEL   FIGURE 1
20INPUTS
  6  12 R        1.00000    0.33333
 12   3 R        0.50000    0.25000
 12  16          0.50000    0.50000
 11  13          1.00000    0.08333
 13  16          1.00000    0.16667
 14  16          1.00000    0.25000
  7  11          0.50000    0.08333
  8  14          1.00000    0.16667
  1   2          1.00000    1.00000
  2   3          0.50000    0.50000
  2   4          0.50000    0.50000
  4   7          0.33330    0.16667
  4   8          0.33330    0.16667
  4   9          0.33330    0.16667
  7  10          0.50000    0.08333
  9  14          0.50000    0.08333
  9  15          0.50000    0.08333
 10  13          1.00000    0.08333
 15  16          1.00000    0.08333
  3   5 R        0.50000    0.33333
  3   6 R        0.50000    0.33333
  5  12 R        1.00000    0.33333
000
   .6          .6          .6          .6          .6          .6          .6          .6
   .6          .6          .6          .6          .6          .6          .6          .6
   .6          .6          .6          .6          .6          .6
```

Card input to program EXP

EXPECTED NUMBER OF ERRORS DETECTED VERSION 1.0

PROGRAM TITLE: ERROR DETECTION-MODEL---FIGURE-1

INPUTS = 20

NUMBER OF ARCS = 22

ARC TAIL	ARC HEAD	BRANCH PROB.	TRAVERSAL PROB.	EXPECTED NUMBER OF ERRORS
6	12	1.0000	0.3333	0.60
12	3	0.5000	0.2500	0.60
12	16	0.5000	0.5000	0.60
11	13	1.0000	0.0833	0.60
13	16	1.0000	0.1667	0.60
14	16	1.0000	0.2500	0.60
7	11	0.5000	0.0833	0.60
8	14	1.0000	0.1667	0.60
1	2	1.0000	1.0000	0.60
2	3	0.5000	0.5000	0.60
2	4	0.5000	0.5000	0.60
4	7	0.3333	0.1667	0.60
4	8	0.3333	0.1667	0.60
4	9	0.3333	0.1667	0.60
7	10	0.5000	0.0833	0.60
9	14	0.5000	0.0833	0.60
9	15	0.5000	0.0833	0.60
10	13	1.0000	0.0833	0.60
15	16	1.0000	0.0833	0.60
3	5	0.5000	0.3333	0.60
3	6	0.5000	0.3333	0.60
5	12	1.0000	0.3333	0.60

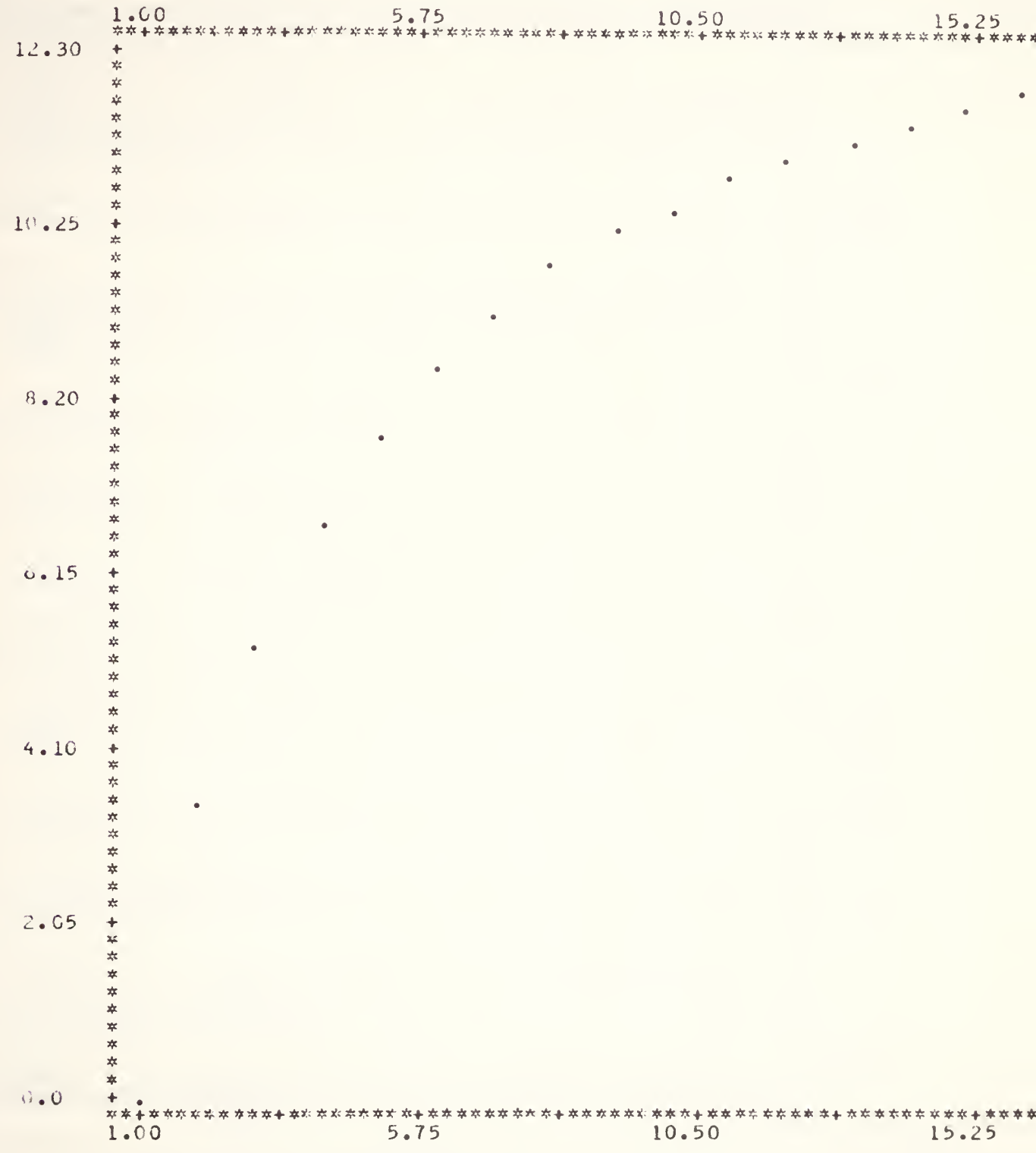
INITIAL EXPECTED NUMBER OF ERRORS= 13.20

INPUT	EXPECTED NUMBER OF ERRORS DETECTED	CUMMULATIVE EXPECTED NUMBER OF ERRORS DETECTED
1	3.45	3.45
2	1.95	5.40
3	1.39	6.79
4	1.04	7.82
5	0.80	8.62
6	0.63	9.25
7	0.51	9.76
8	0.42	10.19
9	0.36	10.54
10	0.30	10.85
11	0.26	11.10
12	0.22	11.33
13	0.20	11.52
14	0.17	11.69
15	0.15	11.84
16	0.13	11.98
17	0.12	12.10
18	0.11	12.20
19	0.09	12.30
20	0.08	12.38

Output of program EXP

GRAPH OF CUMMULATIVE EXPECTED NUMBER OF ERRORS DETECTED -VS- NUMBER OF INPUTS.

PROGRAM TITLE: ERROR DETECTION MODEL FIGURE 1



X-SCALE: "*"= 0.237E 00 UNITS

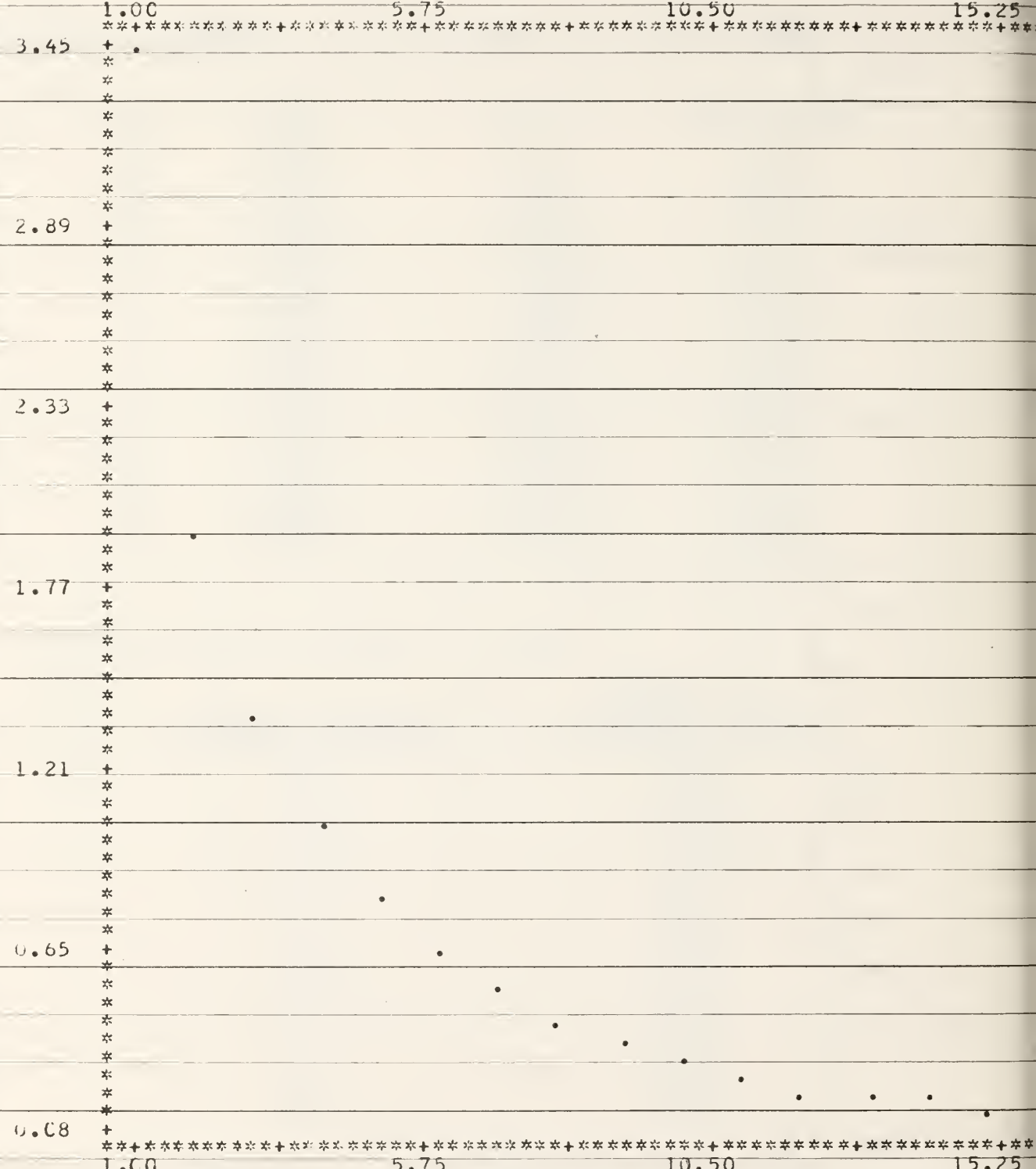
Y-SCALE: "*"= 0.205E 00 UNITS

INPUT NUMBER

Output of program EXP

GRAPH OF EXPECTED NUMBER OF ERRORS DETECTED -VS- NUMBER OF INPUTS.

PROGRAM TITLE: ERROR DETECTION MODEL FIGURE 1



X-SCALE: "*"= 0.237E 00 UNITS

Y-SCALE: "*"= 0.561E-01 UNITS

INPUT NUMBER

Output of program EXP

LIST OF REFERENCES

1. Dijkstra, E. W., "Structured Programming," Report of NATO Conference on Software Engineering Techniques, p. 84-87, 1970.

2. Campbell, D. J. and Heffner, W. J., "Measurement and Analysis of Large Operating Systems During System Developments," Fall Joint Computer Conference, p. 903-914, vol. 33 part 1, 1968.

3. Fishman, G. S., Concepts and Methods in Digital Simulation, p. 262-310, John Wiley & Sons, 1973.

4. Boehm, B. W., McClean, R. K., and Urfrig, D. B., "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software," Proceeding International Conference on Reliable Software, p. 105-113, 1975.

5. Vyssotsky, V. A., "Common Sense in Designing Testable Software," Program Test Methods, p. 41-48, Prentice-Hall, 1973.

6. Poole, P. C., "Debugging and Testing," Advanced Course on Software Engineering, p. 278-318, 1972.

7. Beizer, B., The Architecture and Engineering of Digital Computer Complexes, p. 641-665, Plenum Press, 1971.

8. The Polytechnic Institute of New York, Meaning of Exhaustive Software Testing, by Shooman, M. L., p. 10, January 2, 1974.

9. Schneidewind, N. F., "Analysis of Error Processes in Computer Software," Proceedings International Conference on Reliable Software, p. 337-346, 1975.

10. The Rand Corporation, Software and Its Impact: A Quantitative Assessment, by Boehm, B. W., p. 51, December 1972.

11. The Rand Corporation, Some Information Processing Implications of Air Force Space Missions: 1970-1980, by Boehm, B. W., p. 45, January 1970.

12. Bradley, G. H., Green, T., Howard, G. T. and Schneidewind, N. F.," Structure and Error Detection in Computer Software," Proceedings AIIE National Conference, p. 54-59, 1975.

13. Hetzel, W., "A Definitional Framework," Program Test Methods, p. 7-10 Prentice-Hall, 1973.

14. Dijkstra, E. W., "Notes on Structured Programming," Structured Programming, p. 1-82, Academic Press, 1972.

15. Mills, H., "Top Down Programming in Large Systems," Debugging Techniques in Large Scale Systems, p. 41-53, Prentice-Hall, 1971.

16. Gruenberger, F., "Program Testing: The Historical Perspective," Program Test Methods, Prentice-Hall, p. 11-15.

17. Schneidewind, N.F. and Green, T.F., "Simulation of Error Detection in Computer Programs," Proceedings of the Symposium on the Simulation of Computer Systems, National Bureau of Standards, 1975, six pages.

18. Herbert Y. Chang, et al., Fault Diagnosis of Digital Systems, John Wiley and Sons, 1970.

19. Boehm, B. W., "Software and Its Impact: A Quantitative Assessment," Datamation, v. , p. 48-59, May 1973.

20. Boehm, B. W., "The High Cost of Software," Proceedings of a Symposium on the High Cost of Software, Jack Goldberg (ed), Stanford Research Institute, p. 27-40, 1973.

21. Schneidewind, N.F., "An Approach to Software Reliability Prediction and Quality Control," AFIPS Conference Proceedings, v. 41, Part II, Fall Joint Computer Conference, p. 837-838, 1972.

22. Baker, F. T., "Chief Programmer Team Management of Production Programming," IBM Systems Journal, v. 11, No. 1, p. 65-66, 1972.

23. Rizza, J.B., and Hacker, D., "Quality Assurance Inspection and Test Tools - An Application," Proceedings of a Workshop on Currently Available Program Testing Tools, v. 1, p. 9-10, April 1975.

24. Howden, W. E., "Systems for Automating the Generation of Program Test Data," Proceedings of a Workshop on Currently Available Program Testing Tools, v. 1, p. 37-39, April 1975.

25. Stucki, L.G., "Automatic Generation of Self-Metric Software," Record of 1973 IEEE Symposium on Computer Software Reliability, New York City, p. 94-100, April 30 - May 2, 1973.

26. Von Alven, W. H., (ed), Reliability Engineering, p. 155-156, ARINC Research Corporation, Prentice-Hall, 1964.

27. McCormick, J. M. and Salvadori, M.G., Numerical Methods in FORTRAN, p. 290-295, Prentice-Hall, 1964.

INITIAL DISTRIBUTION LIST

	Copies
Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
Dean of Research Naval Postgraduate School Monterey, California 93940	4
W. R. Church Library Code 0211 Naval Postgraduate School Monterey, California 93940	2
Philip J. Kiviat Technical Director Department of the Air Force AFDAA Center Washington, D. C. 20330	1
Professors G. Howard G. Bradley N. Schneidewind Code 55 Naval Postgraduate School Monterey, California 93940	2 each
Professors J. Esary D. Gaver R. Richards Code 55 Naval Postgraduate School Monterey, California 93940	1 each
Professor U. Kodres Code 72 Naval Postgraduate School Monterey, California 93940	1
Professor M. Powers Code 52 Naval Postgraduate School Monterey, California 93940	1
Professor G. Barksdale Code 72 Naval Postgraduate School Monterey, California 93940	1

Library Code 55 Naval Postgraduate School Monterey, California 93940	1
Knox Library Naval Postgraduate School Monterey, California 93940	2
D. Williams Code 0211 Naval Postgraduate School Monterey, California 93940	1
T. Green	1
G. Montgomery	1

U170550

DUDLEY KNOX LIBRARY - RESEARCH REPORTS
5 6853 01071161 7